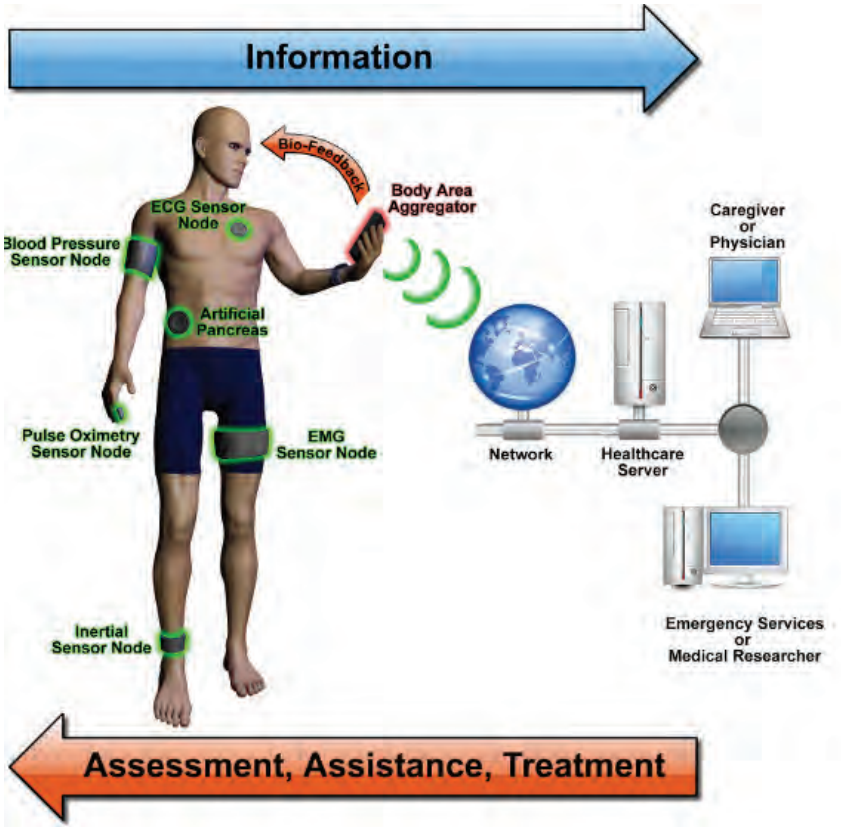
ROYAL INSTITUTE
OF TECHNOLOGY

An Introduction to Wireless Sensor Networks

Draft, version 1.8

Carlo Fischione
September 2014



Contents

List of Acronyms

ACK Acknowledgement

AODV Ad Hoc On-Demand Distance Vector

ASK Amplitude Shift Keying

ARQ Automatic Request Retransmission

AWGN Additive White Gaussian Noise

BER Bit Error Rate

BPSK Bit Phase Shift Keying

CAP Contention Access Period

CCA Clear Channel Assessment

CDF Cumulative Distribution Function

CFP Contention Free Period

CSMA Carrier Sense Multiple Access

CSMA/CA Carrier Sense Multiple Access/Collision Avoidance

CTS Clear-to-Send

FDMA Frequency Division Multiple Access

FFD Full Functionality Device

FSK Frequency Shift Keying

LEACH Low Energy Adaptive Clustering Hierarchy

MAC Medium Access Control

MMSE Minimum Mean Square Error

NLOS Non Line of Sight

PAM	Pulse Amplitude Modulation
PER	Packet Error Rate
PDF	Probability Distribution Function
QPSK	Quadrature Phase Shift Keying
RFD	Reduced Functionality Device
RPL	Routing over Low Power Lossy Networks
RTS	Request-to-Send
SNR	Signal to Noise Ratio
TDMA	Time Division Multiple Access
TDOA	Time Difference of Arrival
TOA	Time of Arrival
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WSN-CS	Wireless Sensor Network-Control Systems

Preface

Recent technological advances led to the development of very small and low-cost sensor devices with computational, processing, data storage and communicational capabilities. These devices, called wireless sensor nodes, when deployed in an area (indoors or outdoors) form a Wireless Sensor Network (WSN). The initial development of WSN was motivated by military applications such as enemy detection, battlefield surveillance, etc. As years went by, considerable amounts of research efforts have enabled the actual implementation and deployment of sensor networks tailored to the unique requirements of certain sensing and monitoring applications. Nowadays WSNs are a very promising tool of monitoring events and are used in many other fields, such as agriculture, environmental monitoring of air-water pollution, greenhouse, health monitoring, structural monitoring and more. Given the benefits offered by WSNs compared to wired networks, such as, simple deployment, low installation cost, lack of cabling, and high mobility, WSNs present an appealing technology as a smart infrastructure for building and factory automation, and process control applications.

The book is intended as a textbook for senior undergraduate or graduate-level students with the goal of helping them to gain an understanding of the challenges and promises of this exciting field. It is also targeted at academic and industrial researchers working in the field, and also at engineers developing actual solutions for WSNs. Each chapter ends with a number of exercises that will allow students to practice the described concepts and techniques.

This book covers fundamental topics to design, understand, and performa a performance analysis of WSNs. The structure is organized as follows:

- Chapter 1 provides an introduction to the basic characteristics and the architecture of a WSN. Moreover, a brief description of the main WSN applications is given;

- Chapter 2 deals with the wireless channel in WSNs. Emphasis is given on the fading models and their effects on the signals that carry the communication messages;

- Chapter 3 presents the physical layer in WSNs. In particular, basic

elements of modulation theory are provided while the probability of error in various channels is studied;

- Chapter 4 covers the medium access control mechanisms in WSNs and the way nodes access the channel is examined. The chapter focuses also on the IEEE 802.15.4 standard;

- Chapter 5 is dedicated to routing in WSNs. Routing protocols are classified, the basic optimization theory for routing is introduced, and an iterative solution for the shortest path optimization problem is presented;

- Chapter 6 presents the fundamental theoretical results for the topology control of WSNs. Emphasis is put on the NP hardness of connectivity and coverage control problems; This is a fairly advanced theoretical chapter.

- Chapter 7 provides an introduction to the basics of detection theory. How events are detected out of uncertain (noisy) measurements from one/multiple sensors is studied;

- Chapter 8 presents the fundamental aspects of distributed estimation over WNSs. Star and ad-hoc networks are studied. Estimation in the presence of limited communication resources is also mentioned. This is a fairly advanced theoretical chapter.

- Chapter 9 introduces the fundamentals of distributed learning over WNSs. After a review of the basics of learning theory, the specific application to WSNs is presented. This is a fairly advanced theoretical chapter.

- Chapter 10 presents the basic of positioning and localization in WSNs. Node positioning methods require the combination of common measurements (e.g. time, range, and angle) together with estimation techniques in order to locate the nodes; This chapter is an application of the results of Chapter 6;

- Chapter 11 introduces the concept of time synchronization and provides an overview of several synchronization strategies; This chapter is an application of the results of Chapter 6;

- Chapter 12 provides an overview of control over WSNs. The basics of automatic control theory are reviewed. Condition ensuring the stability of closed loop control over WSNs are studied, both in the presence of delays and message losses. The effects of WSNs networking protocol is characterized;

- Appendix 1 provides a basic mathematical background for random variables and probability distribution functions;
- Appendix 2 provides a basic mathematical background for sampling theory;
- Appendix 3 gives some basic useful concepts regarding optimization theory;
- Appendix 4 gives one useful result of Matrix Algebra;
- Appendix 5 gives fundamental definitions of Graph Theory;
- Appendix 6 contains an introduction to sensor network programming accompanied with explanatory examples written in NesC, the programming language for WSNs.

This draft book results from the material that has been taught at the 2012 and 2013 editions of the course "Principles of Wireless Sensor Networks" at KTH Royal Institute of Technology, Stockholm, Sweden. The work so far employed to put together this book corresponds to 1.5 years of full time work of one researcher. I acknowledge the work Eric Ahlqvist (Topology Control), Piergiuseppe Di Marco (IEEE 802.15.4 MAC), Charalampos Kalalas (all the chapters, excluded Topology Control and Distributed Learning), Fredrik Isaksson (Distributed Estimation), Gustav Zickert (WSN-Control Systems), Ahsan Mahmood (all the chapters, excluded Topology Control and Distributed Learning), Rasmus Nilsson (Distributed Learning), Yuzhe Xu (Distributed Estimation).

Chapter 1

Introduction to WSNs

Sensor nodes offer a powerful combination of distributed sensing, computing and communication. The ever-increasing capabilities of these tiny sensor nodes, which include sensing, data processing, and communicating, enable the realization of WSNs based on the collaborative effort of a number of other sensor nodes. They enable a wide range of applications and, at the same time, offer numerous challenges due to their peculiarities, primarily the stringent energy constraints to which sensing nodes are typically subjected. As illustrated in Figure 1.1, WSNs incorporate knowledge and technologies from three different fields; Wireless communications, Networking and Systems and Control theory. In order to realize the existing and potential applications for WSNs, sophisticated and extremely efficient communication protocols are required. This chapter provides a first introduction to the WSNs, including architecture, specific characteristics and applications.

1.1 WSN Architecture and Protocol Stack

WSNs, as shown in Figure 1.2, are composed of a number of sensor nodes, which are densely deployed either inside a physical phenomenon or very close to it.

The sensor nodes are transceivers usually scattered in a sensor field where each of them has the capability to collect data and route data back to the sink/gateway and the end-users by a multi-hop infrastructureless architecture through the sink. They use their processing capabilities to locally carry out simple computations and transmit only the required and partially processed data. The sink may communicate with the task manager/end-user via the Internet or satellite or any type of wireless network (like WiFi, mesh networks, cellular systems, WiMAX, etc.), making Internet of Things possible. However, in many cases the sink can be directly connected to the end-users. Note that there may be multiple sinks/gateways and multiple end-users in the architecture.

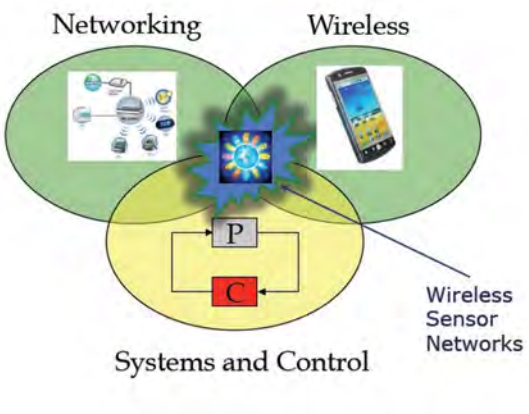


Figure 1.1 Areas of study that concur to the definition of WSNs

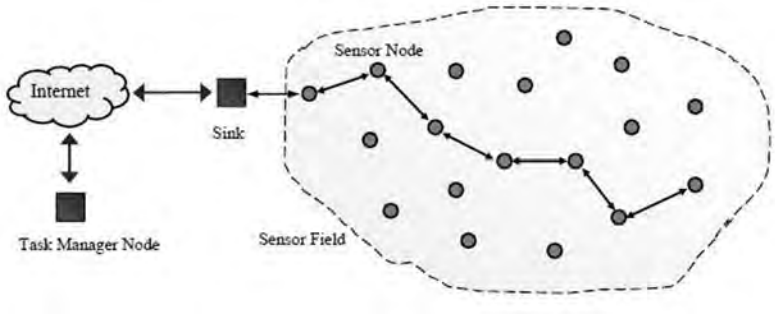


Figure 1.2 A WSN connected to the Internet via a sink node.

As illustrated in Figure 1.3, each sensor node is consisting of five main components; a microcontroller unit, a transceiver unit, a memory unit, a power unit and a sensor unit. Each one of these components is determinant in designing a WSN for deployment.

The microcontroller unit is in charge of the different tasks, data processing and the control of the other components in the node. It is the main controller of the wireless sensor node, through which every other component is managed. The controller unit may consist of an on-board memory or may be associated with a small storage unit integrated into the embedded board. It manages the procedures that enable the sensor node to perform sensing operations, run associated algorithms, and collaborate with the other nodes through wireless communication.

Through the transceiver unit a sensor node performs its communication with other nodes and other parts of the WSN. It is the most power consumption unit.

The memory unit is for temporal storage of the sensed data and can be RAM, ROM and their other memory types (SDRAM, SRAM, EPROM,

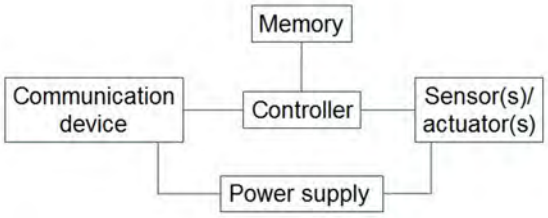


Figure 1.3 Components of a node of a WSN.

etc.), flash or even external storage devices such as USB.

The power unit, which is one of the critical components, is for node energy supply. Power can be stored in batteries (most common) rechargeable or not or in capacitors. For extra power supply and recharge, there can be used natural sources such as solar power in forms of photovoltaic panels and cells, wind power with turbines, kinetic energy from water, etc.

Last but not least is the sensor unit, which is the main component of a wireless sensor node that distinguishes it from any other embedded system with communication capabilities. It may generally include several sensor units, which provide information gathering capabilities from the physical world. Each sensor unit is responsible for gathering information of a certain type, such as temperature, humidity, or light, and is usually composed of two subunits: a sensor and an analog-to-digital converter (ADC). The analog signals produced by the sensor based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the processing unit.

In WSNs, the sensor nodes have the dual functionality of being both data originators and data routers. Hence, communication is performed for two reasons:

- **Source function:** Each sensor node's primary role is to gather data from the environment through the various sensors. The data generated from sensing the environment need to be processed and transmitted to nearby sensor nodes for multi-hop delivery to the sink.

- **Router function:** In addition to originating data, each sensor node is responsible for relaying the information transmitted by its neighbors. The low-power communication techniques in WSNs limit the communication range of a node. In a large network, multi-hop communication is required so that nodes relay the information sent by their neighbors to the data collector, i.e., the sink. Accordingly, the sensor node is responsible for receiving the data sent by its neighbors and forwarding these data to one of its neighbors according to the routing decisions.

Except for their transmit/receive operation state, transceivers can be put

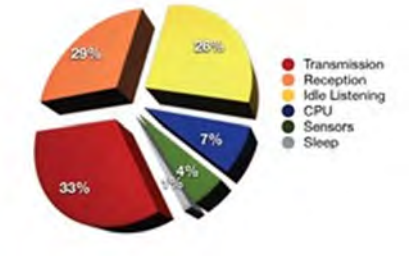


Figure 1.4 Power consumption of a node to receive or transmit messages.

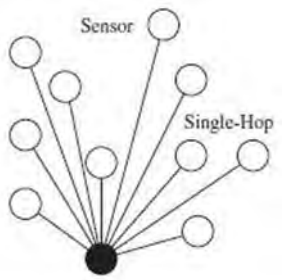


Figure 1.5 WSN having a star topology.

into an idle state (ready to receive, but not doing so) where some functions in hardware can be switched off, reducing energy consumption. The breakdown of the transceiver power consumption in Figure 1.4 shows that a transceiver expends a similar amount of energy for transmitting and receiving, as well as when it is idle. Moreover, a significant amount of energy can be saved by turning off the transceiver to a sleep state whenever the sensor node does not need to transmit or receive any data. In this state, significant parts of the transceiver are switched off and the nodes are not able to immediately receive something. Thus, recovery time and startup energy to leave sleep state can be significant design parameters.

When the transmission ranges of the radios of all sensor nodes are large enough and the sensors can transmit their data directly to the centralized base station, they can form a star topology as shown in Figure 1.5. In this topology, each sensor node communicates directly with the base station using a single hop.

However, sensor networks often cover large geographic areas and radio transmission power should be kept at a minimum in order to conserve energy; consequently, multi-hop communication is the more common case for sensor networks (shown in Figure 1.6). In this mesh topology, sensor nodes must not only capture and disseminate their own data, but also serve as relays for other sensor nodes, that is, they must collaborate to propagate sensor data towards the base station. This routing problem, that is, the task of

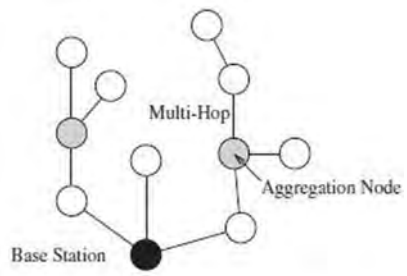


Figure 1.6 WSN having with multihop communication.

finding a multi-hop path from a sensor node to the base station, is one of the most important challenges and has received large attention from the research community. When a node serves as a relay for multiple routes, it often has the opportunity to analyze and pre-process sensor data in the network, which can lead to the elimination of redundant information or aggregation of data that may be smaller than the original data. Routing is examined in detail in chapter 5.

The reduced ISO-OSI protocol stack used by the sink and all sensor nodes is given in Figure 1.7. This protocol stack combines power and routing awareness, integrates data with networking protocols, communicates power efficiently through the wireless medium, and promotes cooperative efforts of sensor nodes. The protocol stack consists of the physical layer, medium access control layer, routing layer and application layer. The physical layer addresses the needs of simple but robust modulation, transmission, and receiving techniques. Since the environment is noisy and sensor nodes can be mobile, the medium access control layer is responsible for ensuring reliable communication through error control techniques and manage channel access to minimize collision with neighbors' broadcasts. The routing layer takes care of routing the data and depending on the sensing tasks, different types of application software can be built and used on the application layer. The above mentioned layers are thoroughly examined in the following chapters.

1.2 Challenges and Constraints

While WSNs share many similarities with other distributed systems, they are subject to a variety of unique challenges and constraints. These constraints impact the design of a WSN, leading to protocols and algorithms that differ from their counterparts in other distributed systems.

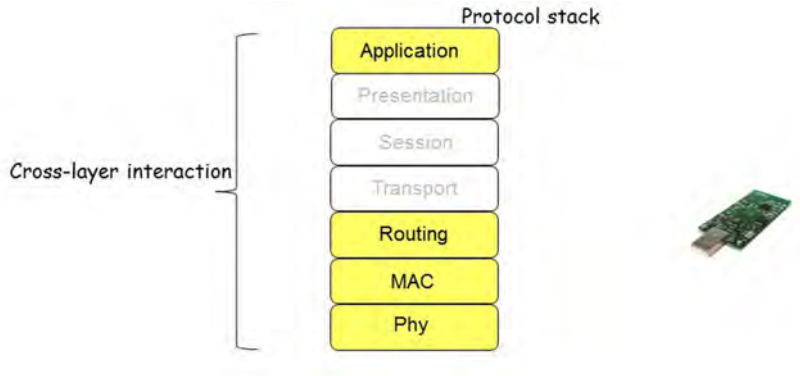


Figure 1.7 ISO-OSI protocol stack for WSNs.

Energy

The intrinsic properties of individual sensor nodes pose additional challenges to the communication protocols primarily in terms of energy consumption. As will be explained in the following chapters, WSN applications and communication protocols are mainly tailored to provide high energy efficiency. Sensor nodes carry limited power sources. Typically, they are powered through batteries, which must be either replaced or recharged (e.g., using solar power) when depleted. For some nodes, neither option is appropriate, that is, they will simply be discarded once their energy source is depleted. Whether the battery can be recharged or not significantly affects the strategy applied to energy consumption. Therefore, while traditional networks are designed to improve performance metrics such as throughput and delay, WSN protocols focus primarily on power conservation.

Node Deployment

The deployment of WSNs is another factor that is considered in developing WSN protocols. The position of the sensor nodes need not be engineered or predetermined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this random deployment requires the development of self-organizing protocols for the communication protocol stack. In particular, sensor nodes must be self-managing in that they configure themselves, operate and collaborate with other nodes, and adapt to failures, changes in the environment, and changes in the environmental stimuli without human intervention. Moreover, many sensor networks, once deployed, must operate unattended, that is, adaptation, maintenance, and repair must be performed in an autonomous fashion. In energy-constrained sensor networks, all these self-management features must be designed and implemented such that they do not incur excessive energy overheads.

Wireless Medium

The reliance on wireless networks and communications poses a number of challenges to a sensor network designer. Large and small-scale fading limit the range of radio signals, that is, a radio frequency (RF) signal attenuates while it propagates through a wireless medium. The received power is proportional to the inverse of the square of the distance from the source of the signal. As a consequence, an increasing distance between a sensor node and a base station rapidly increases the required transmission power. Therefore, it is more energy-efficient to split a large distance into several shorter distances, leading to the challenge of supporting multi-hop communications and routing. Multi-hop communication requires that nodes in a network cooperate with each other to identify efficient routes and to serve as relays.

Hardware Constraints

While the capabilities of traditional computing systems continue to increase rapidly, the primary goal of wireless sensor design is to create smaller, cheaper, and more efficient devices. The five node components described before should also fit into a matchbox-sized embedded system. A sensor's hardware constraints also affect the design of many protocols and algorithms executed in a WSN. For example, routing tables that contain entries for each potential destination in a network may be too large to fit into a sensor's memory. Instead, only a small amount of data (such as a list of neighbors) can be stored in a sensor node's memory. Further, while in-network processing can be employed to eliminate redundant information, some sensor fusion and aggregation algorithms may require more computational power and storage capacities than can be provided by low-cost sensor nodes. Therefore, many software architectures and solutions (operating system, middleware, network protocols) must be designed to operate efficiently on very resource-constrained hardware.

Security

Many wireless sensor networks collect sensitive information. The remote and unattended operation of sensor nodes increases their exposure to malicious intrusions and attacks. Further, the wireless shared medium makes the sensor transmissions insecure. The consequences of a possible intrusion can be severe and depend on the type of sensor network application. Sensor readings must be sent to the sink of the network with a given probability of success, because missing sensor readings could prevent the correct execution of control actions or decisions. However, maximizing the reliability may increase the network energy consumption substantially. While there are numerous techniques and solutions for distributed systems that prevent attacks or contain the extent and damage of such attacks, many of these

incur significant computational, communication, and storage requirements, which often cannot be satisfied by resource-constrained sensor nodes. Hence, the network designers need to consider the tradeoff between reliability and energy consumption and propose new solutions for key establishment and distribution, node authentication, and secrecy.

1.3 WSN Applications

The emergence of the WSN paradigm has triggered extensive research on many aspects of it. The applicability of sensor networks has long been discussed with emphasis on potential applications that can be realized using WSNs. In this section, an overview of certain applications developed for WSNs is provided.

Military or Border Surveillance Applications

WSNs are becoming an integral part of military command, control, communication and intelligence systems. The need of rapid deployment and self-organization characteristics of sensor networks make them a very promising sensing technique for military applications. Since sensor networks are based on the dense deployment of disposable and low-cost sensor nodes, which makes the sensor network concept a better approach for battlefields. Sensors can be deployed in a battle field to monitor the presence of forces and vehicles, and track their movements, enabling close surveillance of opposing forces.

Environmental Applications

The autonomous coordination capabilities of WSNs are utilized in the realization of a wide variety of environmental applications. Some environmental applications of WSNs include tracking the movements of birds, small animals, and insects; monitoring environmental conditions that affect crops and livestock; temperature, humidity and lighting in office buildings; irrigation; large-scale earth monitoring and planetary exploration. These monitoring modules could even be combined with actuator modules which can control, for example, the amount of fertilizer in the soil, or the amount of cooling or heating in a building, based on distributed sensor measurements.

Health Care Applications

Wireless sensor networks can be used to monitor and track elders and patients for health care purposes, which can significantly relieve the severe shortage of health care personnel and reduce the health care expenditures in the current health care systems. For example sensors can be deployed in

a patientâĂŹs home to monitor the behaviors of the patient. It can alert doctors when the patient falls and requires immediate medical attention. In addition, the developments in implanted biomedical devices and smart integrated sensors make the usage of sensor networks for biomedical applications possible.

Home Intelligence

Wireless sensor networks can be used to provide more convenient and intelligent living environments for human beings. For example, wireless sensors can be used to remotely read utility meters in a home like water, gas, electricity and then send the readings to a remote centre through wireless communication. Moreover, smart sensor nodes and actuators can be buried in appliances such as vacuum cleaners, microwave ovens, refrigerators, and DVD players. These sensor nodes inside domestic devices can interact with each other and with the external network via the Internet or satellite. They allow end-users to more easily manage home devices both locally and remotely. Accordingly, WSNs enable the interconnection of various devices at residential places with convenient control of various applications at home.

Industrial Process Control

Networks of wired sensors have long been used in industrial fields such as industrial sensing and control applications, building automation, and access control. However, the cost associated with the deployment and the maintenance of wired sensors limits the applicability of these systems. While sensor-based systems incur high deployment costs, manual systems have limited accuracy and require personnel. Instead, WSNs are a promising alternative solution for these systems due to their ease of deployment, high granularity, and high accuracy provided through battery-powered wireless communication units. Some of the commercial applications are monitoring material fatigue; monitoring product quality; constructing smart office spaces; environmental control of office buildings; robot control and guidance in automatic manufacturing environments; monitoring disaster areas; smart structures with embedded sensor nodes.

Agriculture

Using wireless sensor networks within the agricultural industry is increasingly common; using a wireless network frees the farmer from the maintenance of wiring in a difficult environment. Gravity feed water systems can be monitored using pressure transmitters to monitor water tank levels, pumps can be controlled using wireless I/O devices and water use can be measured and wirelessly transmitted back to a central control center for billing. Irrigation automation enables more efficient water use and reduces waste.

1.4 WSN Integration with the Internet

The evolution of wireless technology has enabled the realization of various network architectures for different applications such as cognitive radio networks, mesh networks, and WSNs. In order to extend the applicability of these architectures their integration with the Internet is very important. So far, research has progressed in each of these areas separately, but realization of these networks will require tight integration and interoperability. In this respect, it is crucial to develop location- and spectrum-aware cross-layer communication protocols as well as heterogeneous network management tools for the integration of WSNs, cognitive radio networks, mesh networks, and the Internet. In this direction, the 6LoWPAN standard has been developed to integrate the IPv6 standard with low-power sensor nodes. Accordingly, the IPv6 packet header is compressed to sizes that are suitable for sensor motes. This provides efficient integration for communication between an IPv6-based device and a sensor mote. However, significant challenges in seamless integration between WSNs and the Internet still exist at the higher layers of the protocol stack. The coexistence of WLANs and WSNs is a major challenge at the MAC layer since they both operate in the same spectrum range. End-to-end routing between a sensor node and an Internet device is not feasible using existing solutions. Similarly, existing transport layer solutions for WSNs are not compatible with the TCP and UDP protocols, which are extensively used in the Internet. In most sensor deployment scenarios, the sink is usually assumed to reside within or very near to the sensor field, which makes it part of the multi-hop communication in receiving the sensor readings. However, it would be desirable to be able to reach the sensor network from a distant monitoring or management node residing in the wireless Internet. Therefore, new adaptive transport protocols must be developed to provide the seamless reliable transport of event features throughout the WSN and next-generation wireless Internet. Moreover, Internet protocols are generally prone to energy and memory inefficiency since these performance metrics are not of interest. Instead, WSN protocols are tailored to provide high energy and memory efficiency. The fundamental differences between the design principles for each domain may necessitate novel solutions that require significant modifications in each network to provide seamless operation.

Problems

Problem 1.1 Gaussian Q function

(a) Consider a random variable X having a Gaussian distribution with zero mean and unit variance. The probability that X is larger than x, or distribution

function, is

$$\mathbf{P}(X > x) = Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt\,,$$

where $Q(\cdot)$ is called the Q function. Plot the distribution function in the variable x. Recalling that a function is convex when the second derivative is strictly positive, find a region of x in which the function is convex.

(b) Consider a Gaussian random variable $X \sim \mathcal{N}(\mu, \sigma)$ of average μ and standard deviation σ. Such a random variable has a distribution function given by a translated and reshaped Q function:

$$Q\left(\frac{x-\mu}{\sigma}\right).$$

Discuss about convexity region of this function.

(c) A function f is log-concave if $f(x) > 0$ and for all x in its domain $-\log f(x)$ is convex. Show that the twice differentiable function Q is log-concave.

Problem 1.2 Binary hypothesis testing: application of the Q function

Assume a couple of sensor nodes are randomly deployed in a region of interest and are connected to a sink. The task of each sensor is to detect if an event happened or not, namely taking a binary decision. Each sensor measures noisy signals from the environment and whenever the measured signal is strong enough the sensor will decide that an event has occurred. We assume that the measurement noises at sensor i are identically and independently distributed (i.i.d) and follows a Gaussian distribution $n_i \sim \mathcal{N}(0, 1)$. The binary hypothesis testing problem for sensor i is as follows:

$$H_1 : s_i = a_i + n_i$$

$$H_0 : s_i = n_i\,,$$

where s_i is the measured signal at sensor i, and $a_i \in \mathbb{R}_+$ is the signal amplitude associated to the event. Assume that all sensors use a common threshold τ to detect the event, i.e., if the measured signal at sensor i is larger than τ, then the sensor will decide that the event happened and will report this decision to the sink.

(a) Characterize the probability of *false alarm* p_f, namely the probability that a local sensor decides that there was an event while there was not one.

(b) Characterize the probability of *detecting* an event p_d, namely the probability that an event occurs and the sensor detects it correctly.

Problem 1.3 Miscellanea of discrete random variables (Ex. 3.24 in (Boyd and Vandenberghe, 2004))

Let X be a real-valued random variable that takes discrete values in $\{a_1, a_2 \ldots, a_n\}$ where $a_1 < a_2 < \cdots < a_n$, with probability $\mathbf{P}(X = a_i) = p_i, \forall i = 1, 2, \ldots, n$. Characterize each of following functions of $\mathbf{p} = [p_i]$ $\{\mathbf{p} \in \mathbb{R}_+^n | \mathbf{1}^T \mathbf{p} = 1\}$ (where $\mathbf{1}$ is the all ones vector) and determine wether the function is convex or concave.

(a) Expectation: $\mathbf{E}X$.

(b) Distribution function: $\mathbf{P}(X \geq \alpha)$.

(c) Probability of interval: $\mathbf{P}(\alpha \leq X \leq \beta)$.

(d) Negative entropy distribution: $\sum_{i=1}^{n} p_i \log p_i$.

(e) Variance: $\mathbf{var}X = \mathbf{E}(X - \mathbf{E}X)^2$.

(f) Quartile: $\mathbf{quartile}(X) = \inf\{\beta \,|\, \mathbf{P}(X \leq \beta) \geq 0.5\}$.

Problem 1.4 Amplitude quantization

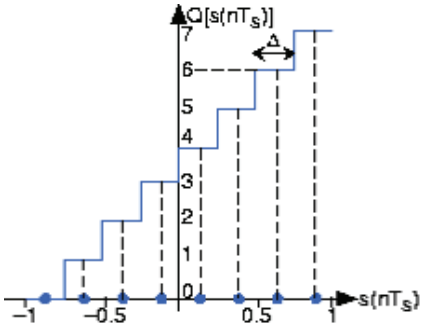
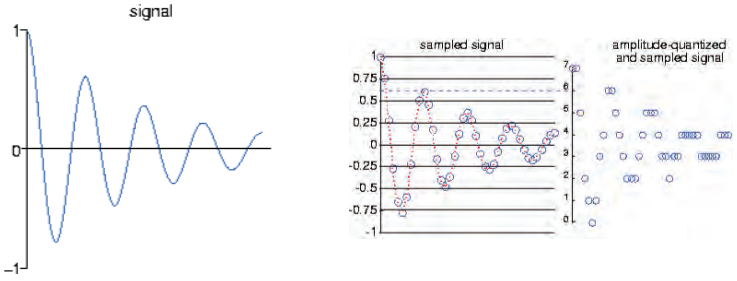


Figure 1.8 (a)A three-bit Analog to Digital (A/D) converter assigns voltage in the range $[-1, 1]$ to one of eight integers between 0 and 7. For example, all inputs having values lying between 0.5 and 0.75 are assigned the integer value six and, upon conversion back to an analog value, they all become 0.625. The width of a single quantization interval Δ is $2/2^B$.

The analog-to-digital (A/D) conversion is a standard operation performed in sensors and many electronic devices. Itworks as follows: Consider a sensor that samples a bandlimited continuos time signal $s(t)$. According to sampling theory, if the sensor samples the signal fast enough at time nT_s, where n is the sample number and T_s is the sampling time, it can be recovered without error from its samples $s(nT_s)$, $n \in \{\ldots, -1, 0, 1, \ldots\}$. The processing of the data further requires that the sensor samples be quantized: analog values are converted into digital form. The computational round-off prevents signal amplitudes from being converted with no errors into a binary number representation.

In general, in A/D conversion, the signal is assumed to lie within a predefined range. Assuming we can scale the signal without affecting the information it expresses, we will define this range to be $[-1, 1]$. Furthermore, the A/D converter assigns amplitude values in this range to a set of integers. A B-bit converter produces one of the integers $\{0, 1, \ldots, 2^B - 1\}$ for each sampled input. Figure 1.8 shows how a three-bit A/D converter assigns input values to the integers. We define a quantization interval to be the range of values assigned to the same integer. Thus, for our example three-bit A/D converter, the quantization interval Δ is 0.25; in general, it is $2/2^B$.

Since values lying anywhere within a quantization interval are assigned the same value for processing, the original amplitude value is recovered with errors. The D/A converter, which is the device that converts integers to amplitudes, assigns an amplitude equal to the value lying halfway in the quantization interval. The integer 6 would be assigned to the amplitude 0.625 in this scheme. The error introduced by converting a signal from analog to digital form by sampling and amplitude quantization then back again would be half the quantization interval for each amplitude value. Thus, the so-called A/D error equals half the width of a quantization interval: $1/2^B$. As we have fixed the input-amplitude range, the more bits available in the A/D converter, the smaller the quantization error.



(a) (b)

Figure 1.9 (a) Shows a signal going through the analog-to-digital, where B is the number of bits used in the A/D conversion process (3 in the case depicted here). First it is sampled (b), then amplitude-quantized to three bits. Note how the sampled signal waveform becomes distorted after amplitude quantization. For example the two signal values between 0.5 and 0.75 become 0.625. This distortion is irreversible; it can be reduced (but not eliminated) by using more bits in the A/D converter.

To analyze the amplitude quantization error more deeply, we need to compute the signal-to-noise ratio, which is the ratio of the signal power and the quantization error power. Assuming the signal is a sinusoid, the signal power is the square of the root mean square (*rms*) amplitude: power(s) $= (1/\sqrt{2})^2 = 1/2$. Figure 1.9 shows the details of a single quantization interval.

Its width is Δ and the quantization error is denoted by ϵ. To find the power in the quantization error, we note that no matter into which quantization interval the signal's value falls, the error will have the same characteristics. To calculate the *rms* value, we must square the error and average it over the interval.

$$\mathrm{rms}(\epsilon) = \sqrt{\frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \epsilon^2 \, \mathrm{d}\epsilon} = \left(\frac{\Delta^2}{12}\right)^{1/2}$$

Since the quantization interval width for a B-bit converter equals $2/2^B = 2^{1-B}$, we find that the signal-to-noise ratio for the analog-to-digital conversion process equals

$$\mathrm{SNR} = \frac{\frac{1}{2}}{\frac{2^{2(1-\mathrm{B})}}{12}} = \frac{3}{2}2^{2\mathrm{B}} = 6\mathrm{B} + 10\log 1.5\mathrm{dB}$$

Thus, every bit increase in the A/D converter yields a 6 dB increase in the signal-to-noise ratio. The constant term $10 \log 1.5$ equals 1.76.

(a) This derivation assumed the signal's amplitude lay in the range $[-1, 1]$. What would the amplitude quantization signal-to-noise ratio be if it lay in the range $[-A, A]$?

(b) How many bits would be required in the A/D converter to ensure that the maximum amplitude quantization error was less than 60 db smaller than the signal's peak value?

(c) Music on a CD is stored to 16-bit accuracy. To what signal-to-noise ratio does this correspond?

Problem 1.5 Accelerometer system design and system scale estimate (Ex.4.1 in (Pottie and Kaiser, 2005))

An accelerometer is a sensor that measures acceleration. Consider the design of an accelerometer that is intended to meet specific acceleration sensitivity goals over a specific bandwidth given a position sensor sensitivity. The designer may adjust mass, spring constant, proof mass value, and resonance quality factor to achieve these goals.

(a) Consider an accelerometer with an electronic displacement sensor having a position sensitivity of $1\text{pm}/(\text{Hz})^{1/2}$. For a target acceleration sensitivity of 10^{-5} $\text{m/s}^2/(\text{Hz})^{1/2}$ in the bandwidth from 0.001 to 100 Hz, find the largest sensor resonance frequency that may meet this objective while ignoring the effect of thermal noise.

(b) Now, include the effect of thermal noise and compute the required proof mass value for this accelerometer for Q values of 1, 100, and 10^4 (consider parameters $K_b = 1.38 \times 10^{-23}$ and $T = 300$).

(c) If this mass were to be composed of a planar Si structure, of thickness 1μ, what would be the required area of this structure.

Problem 1.6 Signal dependent temperature coefficients (Ex.4.4 in (Pottie and Kaiser, 2005))

A silicon pressure microsensor system employs a piezoresistive strain sensor for diaphragm deflection having a sensitivity to displacement of $\alpha = 1\text{V}/\mu$ (at $T = 300\text{K}$). Further, this displacement is related to pressure with a pressure-dependent deflection of $\text{K} = 0.01\mu/\text{N/m}^2$. This is followed by an amplifier having a gain $G = 10$ (at $T = 300K$). This amplifier further shows an input-referred offset potential, $V_{\text{offset}} = 0$ at 300K. Each of these characteristics include temperature coefficients. These temperature coefficients are listed here:

α	$10^{-2}/\text{K}$
K	$10^{-4}/\text{K}$
G	$-10^{-3}/\text{K}$
V_{offset}	$-10\mu\text{V/K}$

(a) Consider that the pressure sensor is exposed to no pressure difference. Find an expression for its output signal for temperature. Compute the temperature coefficient that describes the operation.

(b) Consider that the pressure sensor is exposed to a pressure difference signal of 0.1 N/m^2. Find an expression for its output signal for temperature and plot this. Estimate the temperature coefficient that describes its operation at the specific temperatures in the neighborhood of 250K and 350K.

(c) Consider that the pressure sensor is exposed to a pressure difference signal of 10 N/m^2. Find an expression for its output signal for temperature and plot this. Estimate the temperature coefficient that describes its operation at the specific temperatures in the neighborhood of 250K and 350K.

Chapter 2

Wireless Channel

The wireless channel introduces important restrictions to the performance of WSNs. Simply speaking, the wireless channel has an important role in determining the distance to which a message can be transmitted from a sensor node, and the probability of receiving successfully the message at some receiver node.

The transmission of messages is performed by an electromagnetic wave transmitted by the antenna of the sender node. The power of the electromagnetic waves are received at the antenna of the receiver node distorted and attenuated due to the wireless propagation that is subject to several external factors. The result is that the power of the waves may be so attenuated that the wave signal cannot be correctly detected at the receiver node. More specifically, the transmitted wave signal undergoes attenuations while traveling over the wireless channel through the propagation path from the transmitter to the receiver node. The effect of these attenuations is commonly called fading. In free space propagation, namely a propagation of the wave signal from the transmitter to the receiver without any obstacle in between, the wave signal arrives at the receiver by a constant attenuated power due to the "path loss", which we will see later. However, when a signal encounters obstacles in the propagation path from the transmitter to the receiver, or the signal is reflected by obstacles or reflectors, the attenuation is no longer just constant and follows a more complex law due to the physics of the wireless channel. Here, the signal is reflected, diffracted, and scattered from objects that are present in the path. Each path can have a different amount of attenuation, delay and fading amount. The combination of these different paths is termed multipath fading or multipath propagation. At the receiver, the waves signals coming from different reflections can add constructively or destructively, causing random and rapid fluctuations in the received power at the receive antenna, especially when the receiver or the transmitter is moving. Due to the Doppler effect, this situation also causes the signal to be spread in the frequency domain (fad, 2020), meaning that

a wave signal sent offer a certain carrier frequency is received over shifted frequencies.

The stochastic character of the wireless channel is a factor that affects severely the signal propagation creating a time-variant environment. All these effects result in erroneous decoding of the transmitted wireless wave at the antenna of the receiver node. This is often called "wireless channel errors". Understanding the effects of the wireless channel errors can be done by a mathematical modeling of the attenuations that the transmitted wave signals undergo over the channel. Since the adverse effects of these errors influence each protocol in WSNs, the examination of the effects of the wireless channel is essential in WSNs.

This chapter is organized as follows: The next section briefly discusses the dominant sources of attenuation in the wireless channel and summarizes the main fading factors that occur in signal propagation during the sensors' communication. Then a mathematical modeling of the large-scale fading is presented, followed by the modeling of the small-scale fading phenomena. The last section outlines the conclusion of the chapter.

2.1 Physical Sources of Distortion

The wireless channel distorts signals transmitted from a transmitter node. The cause of this distortion can be classified into four main phenomena (Akyildiz and Vuran, 2010):

2.1.1 Attenuation (Path Loss)

The term refers to the reduction in power density (attenuation) of the electromagnetic wave as it propagates through space as function of the distance. The attenuation is proportional to the distance travelled by the wave over the space.

2.1.2 Reflection and refraction

When a signal wave is incident at a boundary between two different types of material, a certain fraction of the wave is absorbed by the material, whereas another fraction bounces off the surface, which is called reflection. Depending on the properties of the two materials, a certain fraction of the wave may also propagate through the boundary, which is called refraction. Reflection and refraction are usually observed on the ground or the walls of a building as shown in Figure 2.1(a). More generally, these phenomena occur in case of obstructing objects with large dimensions compared to the wavelength. As a result the signal received at the antenna of the receiver node may fade based on the constructive or destructive effects of multiple waves that are received.

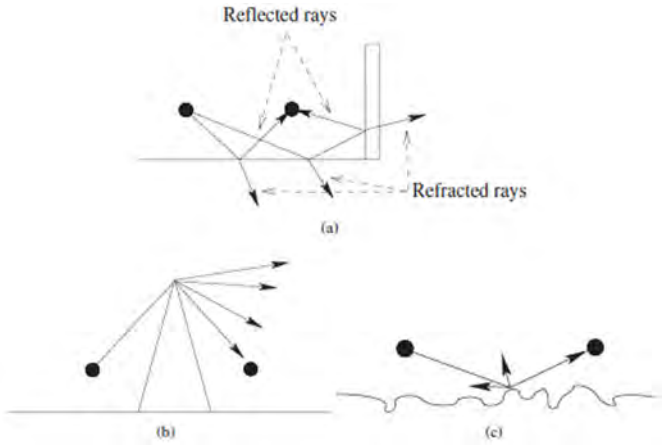


Figure 2.1 Propagation phenomena of signals over the wireless channel (Akyildiz and Vuran, 2010).

2.1.3 Diffraction

The term refers to the phenomena that occur when an electromagnetic wave propagates through sharp edges such as the tip of a mountain or a building or surfaces with irregularities. As shown in Figure 2.1(b), this causes the sharp edge to act as a source, where new secondary waves are generated giving rise to a bending of waves around and behind the obstacle. In effect, the original signal strength is distributed to the new generated waves.

2.1.4 Scattering

Signal waves do not generally encounter obstacles with perfect boundaries. Instead, when a signal wave is incident at a rough surface, it scatters in random directions as shown in Figure 2.1(c). This phenomenon is also encountered in case of a radio wave traveling through a medium containing many small (compared to the wavelength) objects, which influence the propagation.

2.2 Statistical fading models

The physical causes of fading of a transmitted signal wave can be modeled statistically. These statistical mathematical models are very useful to characterize the probability to receive messages transmitted over the wireless channels. The availability of these statistical models allow to tie the probability to successful message reception with the characteristics of the wireless channel, the transmit radio power, and many other parameters such as modulation, coding, etc.

According to Friis transmission equation, the received power can be written as (Sli, 2013):

$$P_r = P_t G_t(\theta_t, \psi_t) G_r(\theta_r, \psi_r) \frac{\lambda^2}{(4\pi r)^2} \overline{\mathrm{PL}} z\, y\,, \tag{2.1}$$

where $G_t(\theta_t, \psi_t)$ is the transmit antenna gain in the direction (θ_t, ψ_t), $G_r(\theta_r, \psi_r)$ is the receive antenna gain in the direction (θ_r, ψ_r), $\lambda^2/(4\pi r)^2 \overline{\mathrm{PL}}\, y$ is the large scale fading, and z is the small scale fading. In particular, large scale fading refers to the path loss describing the variation of the attenuation with respect to the distance (the term $\lambda^2/(4\pi r)^2 \overline{\mathrm{PL}}$), and the shadow fading (the term y), which, in turn, describes the random variation of the attenuation for a specific distance. On the other hand, small scale fading z refers to the abrupt changes in signal amplitude and phase that can be experienced as a result of small changes (as small as half wavelength) in the spatial position between transmitter and receiver (Pantos et al., 2008). These fluctuations are due to the signal's time spread caused by multipath propagation and due to the channel's variation in time because of the movements of the nodes and scattering objects that participate in the propagation. In the following sections, a mathematical modeling of large-scale and small-scale phenomena are presented.

2.3 Large Scale Fading

2.3.1 Path Loss

Path loss is the attenuation in signal power (strength) of the signal wave as it propagates through air. Path loss is proportional to the distance between the transmitter and the receiver. The causes for path loss are free space loss, refraction, reflection, diffraction, absorption and others. The signal strength decreases with distance and when it is below threshold (receiver sensitivity) the distance is called maximum communication range of the transmitter. In the received power expression 2.1, the path loss is

$$\mathrm{PL} = \frac{(4\pi r)^2}{\lambda^2} \overline{\mathrm{PL}}\,. \tag{2.2}$$

Generally, the path loss can be represented as the ratio of the transmitted power at a node, P_t and the received power, P_r . In logarithmic scale,

$$\mathrm{PL}(d)[\mathrm{dB}] = \mathrm{PL}(d_0)[\mathrm{dB}] + 10n \log_{10}\left(\frac{d}{d_0}\right), \tag{2.3}$$

where $\mathrm{PL}(d_0)[\mathrm{dB}]$ is the path loss at the reference distance, d_0, in dB, $\mathrm{PL}(d)[\mathrm{dB}]$ is the path loss at distance, d, in dB and n is the path loss exponent. It can be seen from the above mentioned equation that path loss

increases with the communication distance; that means we need to increase transmitted power level for successful communication at longer distances. In addition, path loss also increases with the carrier frequency; that explains why higher carrier frequencies suffer from higher propagation losses. The path loss exponent n depends on the topology of the terrain and on the characteristic of the medium, such as oxygen concentration and temperature. Normally, its value ranges from $2 - 6$. For free space, it has a value of 2 and it is equal to 4 for the plane earth model.

Path loss can be mathematically modeled by different models such as

1. Physical path loss models
   (a) Power law propagation models
   (b) Ray tracing
2. Empirical path loss models
   (a) Okumura and Hata's mode
   (b) COST-231 - Hata model

These mathematical models have been heavily investigated in the literature.

2.3.2 Shadowing

Radio signals are often shadowed while transmitted by buildings or other large obstacles resulting in an Non Line Of Sight (NLOS) path between the transmitter and the receiver. Shadow fading is a phenomenon that occurs when a node moves behind an obstruction (sized hundreds times of wavelength λ) and experiences a significant reduction in signal power. The received signal power can be modeled by a random variable that depends on the number and the characteristics of the obstructing objects that are located in the propagation area and participate in the process of signal propagation. Therefore, the value of the received power may differ substantially from the path loss model (Pantos et al., 2008). The path loss can be seen as the statistical mean value of the received power.

Based on radio channel measurements, the shadow fading in the received signal power expressed in logarithmic scale (dBm or dBW) follows a Gaussian distribution, with its mean value being determined by the path loss exponent n and standard deviation σ that depends on the environment. Thus, the received power as function of the path loss and shadowing is extended as (Akyildiz and Vuran, 2010)

$$\mathrm{PL}(d)[\mathrm{dB}] = \mathrm{PL}(d_0)[\mathrm{dB}] + 10n \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma\,, \tag{2.4}$$

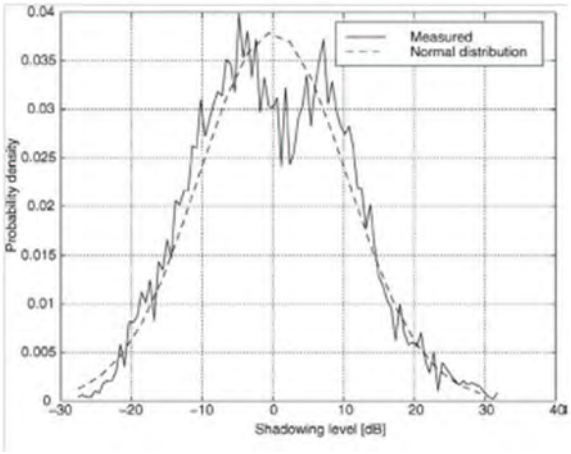


Figure 2.2 PDF for wireless channels with Lognormal attenuations (Pantos et al., 2008).

where X_σ is random variable (in dB) that follows a Gaussian distribution with zero mean and variance σ^2. Thus in linear units, the statistical distribution of the wireless channel undergoing shadow fading is Lognormal. The standard deviation of the shadowing is called as "location variability" and varies with the frequency, the antenna heights and the environment. The probability density function of shadowing is depicted in Figure 2.2.

A physical explanation for Lognormal distribution is given as follows. Considering the contributions A_i to the signal attenuation along the propagation path acting independently, then the total attenuation A due to N individual contributions will be simply the product of the contributions

$$A = A_1 \cdot A_2 \cdot ... \cdot A_N \,, \tag{2.5}$$

and in dB scale, the sum of the individual losses

$$L = L_1 + L_2 + ... + L_N \,. \tag{2.6}$$

If all of the contributions L_i are taken as random variables, then according to the Central Limit Theorem, L is a Gaussian random variable and, therefore, A must be lognormal. In practice, not all of the losses will contribute equally, with those nearest the receiver node being most harmful and the contributions of individual diffracting obstacles cannot simply be added, so the assumption of independence is not strictly valid. However, when the different building heights, spacing and construction methods are taken into account, along with the attenuation due to trees, the resultant distribution function is very close to lognormal (Pantos et al., 2008).

2.4 Small Scale Fading

2.4.1 Multipath Fading

Multipath is defined as the propagation phenomenon that results in radio signals reaching the receiving node by multiple paths. Multipath propagation occurs because of the presence of physical objects that lead signals to be reflected and scattered. It can contribute to a constantly changing environment in the channel that dissipates signal energy in amplitude, phase and time. These multiple replicas of the transmitted signal that arrive in the receiver produce random phase and amplitude, which result in fluctuations in signal strength. Based on the environment multipath fading can follow different distributions; the most common ones are reported in the following subsections.

Rayleigh Fading

Rayleigh fading is a statistical model that is often used to describe the effect of a propagation environment on a radio signal due to scattering. It is most applicable when there is no dominant propagation along a line of sight between the transmitting and receiving node. Because there is no direct ray component, Rayleigh fading is often classified as the worst case fading type (Pottie and Kaiser, 2005).

Rayleigh fading models assume that the complex envelope $c(t)$ of the received signal is the sum of many random complex components arriving from different paths. Its amplitude $r(t)$ follows the Rayleigh distribution (Pantos et al., 2008), namely

$$c(t) = x(t)e^{\theta(t)}\,, \tag{2.7}$$

where, recalling z in (2.1),

$$x(t) \triangleq \sqrt{z(t)} = \sqrt{[I(t)]^2 + [Q(t)]^2}\,,$$

$$\theta(t) = \arctan\left(\frac{Q(t)}{I(t)}\right)\,,$$

and $I(t), Q(t)$ are the baseband orthogonal components of the received passband signal which are given by

$$I(t) = \sum_{i=1}^{N} a_i \cos(\omega_i(t) + \psi_i)\,, \tag{2.8}$$

$$Q(t) = \sum_{i=1}^{N} a_i \sin(\omega_i(t) + \psi_i)\,, \tag{2.9}$$

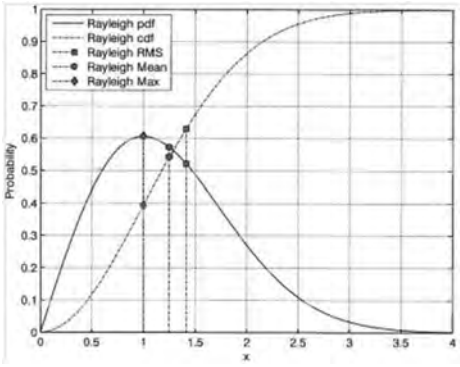


Figure 2.3 Rayleigh PDF and CDF (Pantos et al., 2008).

where a_i and ψ_i are random variables that represent the amplitude and phase respectively of each received component. If the number N of the received components is larger than 6, then the Central Limit Theorem exists and $I(t), Q(t)$ are independent Gaussian random variables with zero mean value and variance equal to σ^2 (Pantos et al., 2008; Pottie and Kaiser, 2005). Thus, the amplitude $x(t)$ follows the Rayleigh distribution with probability density function given by

$$p_r(x) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \, , \; 0 \leq r \leq \infty \, . \tag{2.10}$$

In Figure 2.3, the Rayleigh distribution and its characteristic values are presented.

Rician Fading

In case there is a strong component (usually an LOS component or a strong reflection) in addition to the other multipath components, then the amplitude and phase distributions of the complex envelope are different from the previous case. The complex envelope has now the following form

$$c(z) = c_0 + \sum_{i=1}^{N} c_i(z) \, . \tag{2.11}$$

Real and imaginary part of $c(z)$ remain Gaussian with the same variance but now their mean values are not equal to zero (Pantos et al., 2008). Recalling z in (2.1), and letting $x^2 = z$, the amplitude of the complex envelope follows Rician distribution whose probability density function is given by

$$p_r(x) = \frac{r}{\sigma^2} e^{-\frac{x^2+|c_0|^2}{2\sigma^2}} I_0\left(\frac{x|c_0|}{\sigma^2}\right) . \tag{2.12}$$

where

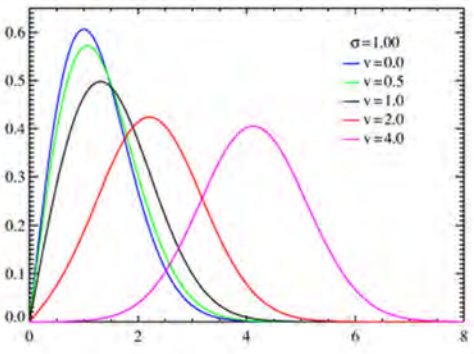


Figure 2.4 Ricean PDF (Ric, 2013).

$$I_0(x) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{x\cos(\theta)} d\theta$$

is the modified Bessel function of order zero of the first kind. Figure 2.4 shows the probability density function for Rician distribution for different values of $|c_0|$ (parameter v denotes $|c_0|$).

The ratio between the power in the direct path and the power in the other, scattered, paths is called Rician K factor and is defined as (Pantos et al., 2008):

$$K = \frac{|c_0|^2}{2\sigma^2}\,, \tag{2.13}$$

$$K(\text{dB}) = 10\log_{10}\left(\frac{|c_0|^2}{2\sigma^2}\right)\,. \tag{2.14}$$

From the previous definition, it can be concluded that Rayleigh fading is the specialized model for stochastic fading when there is no line of sight signal and is sometimes considered as a special case ($K = 0$) of the more generalized concept of Rician fading. For large values of K, the distribution approximates Gaussian with mean value equal to c_0.

Nakagami Fading

Another useful type of distribution is Nakagami-m which has similar behavior to the Rician one. If the Central Limit Theorem is not satisfied, then Nakagami-m is an approximate distribution for the amplitude of the complex envelope (Pantos et al., 2008). This distribution has more general application since it is used in order to describe either better or worse fading conditions than Rayleigh and Rician distribution by choosing properly the value of parameter m. More specifically, the Nakagami-m distribution models very well the distribution of signal envelopes in a variety of fading

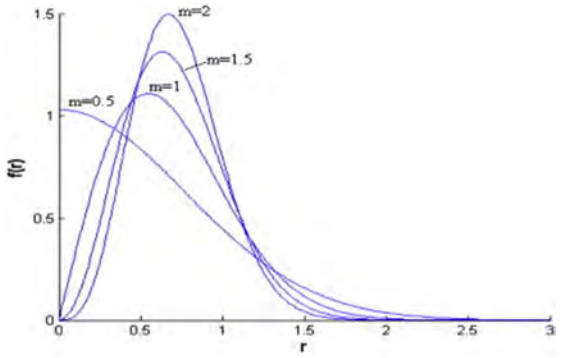


Figure 2.5 Nakagami-m PDF.

environments, ranging from a strong line-of-sight environment to a highly diffuse scattering environment. The degree of fading in this distribution is characterized by the shape parameter m, which takes a value greater than or equal to half, where $m = 1/2$ corresponds to one-sided Gaussian fading (it represents the maximum amount of fading that the Nakagami-m distribution can characterize), $m = 1$ and $1 < m < \infty$ corresponds to Rayleigh and Rician fading respectively. An infinite m corresponds to a deterministic envelope (it represents the case of no fading) (Kallik, 2010). Thus the degree of fading decreases with increase of the parameter m. Recalling that $x = \sqrt{z}$, the probability density function of Nakagami-m is given by

$$p_r(x) = \frac{2}{\Gamma(m)} \left(\frac{m}{\Omega}\right)^m x^{2m-1} e^{-\frac{m}{\Omega}x^2}, \qquad x \geq 0, m \geq \frac{1}{2}. \tag{2.15}$$

where $\Omega = E[x^2]$ is the spread of the distribution, and

$$m = \frac{\Omega^2}{(x^2 - \Omega)^2}$$

is the shape parameter and $\Gamma(m)$ is the gamma Euler function. Figure 2.5 shows the above mentioned pdf for different values of parameter m. Finally, the relation between the shape parameter m and the Rician K factor are given as (Pantos et al., 2008):

$$m = \frac{(K+1)^2}{2K+1}, \tag{2.16}$$

$$K = \frac{\sqrt{m^2 - m}}{m - \sqrt{m^2 - m}}. \tag{2.17}$$

2.4.2 Doppler Spread

Due to the relative motion between the nodes that are in the communication link, each multipath wave experiences an apparent shift in frequency.

The shift in received signal frequency due to motion is called the Doppler spread, and is directly proportional to the velocity and direction of motion of the mobile node with respect to the direction of the arrival of the received multipath wave. This phenomenon occurs during a transmission where a receiver node moves towards or away from a transmitter node. Objects in the radio channel will induce a time varying Doppler spread on multipath components if they are in motion. This means that if the surrounding objects move at a greater rate than the receiver, the effect dominates the small-scale fading and otherwise the motion may be ignored and only the speed of the receiver needs to be considered (Pantos et al., 2008).

2.5 Conclusion

In this chapter the impact of different attenuation effects that occur in the wireless channel was summarized. A study of the different fading phenomena was performed together with a mathematical description based on the statistical properties of each phenomenon. The effects which were discussed can help in calculating receiver sensitivity and link budget analysis for the design of wireless sensor networks. The mathematical modeling of the wireless channel is very useful to understand the probability of message losses when transmitting messages over the wireless channel, as it will be studied in next chapter. Therefore, since all the previously examined factors result in an overall degradation in the performance of communication over a WSN, wireless channel attenuation should be highly considered for applications that involve wireless sensors.

Problems

Problem 2.1 The noisy sensor (Ex.14.6 in (Pottie and Kaiser, 2005))

Sensor nodes are laid out on a square grid of spacing d as reported in Figure 2.6. For simplicity, propagation losses go as the second power of distance. The source to be detected has a Gaussian distribution with zero mean and variance σ_n^2. The source is measured at each sensor by a noisy measurement having an independent Additive White Gaussian Noise (AWGN) with variance σ_S^2 . Sensor node 1 is malfunctioning, producing noise variance $10\sigma_n^2$. The two best nodes in terms of SNR cooperate to provide estimates of the source.

(a) Sketch the region of source locations over which node (1) will be among the two best nodes, assuming a long sequence of measurements are made of the source.

(b) For a single measurement, approximate the likelihood that a source at position $(0.25d, 0)$ will result in better SNR at sensor 5 than at sensor 1.

Problem 2.2 Radio power optimization

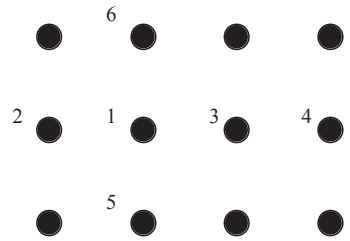


Figure 2.6 A sensor network.

Consider the following model describing the required energy $E(A, B)$ to send a packet from node A to node B: $E(A, B) = d(A, B)^\alpha$. Here, $d(A, B)$ is the distance between node A and B and α is a system parameter with $\alpha > 2$. Assume that we are allowed to place a number of equidistant relay nodes between source node S and destination node T. Here, relay nodes serve as intermediate nodes to route packets from S to T. For instance, if S and T would use relay nodes A and B, the message would be sent from S to A, from A to B and finally from B to T.

(a) What is the ideal number of relay nodes in order to send a message from S to T with minimum energy consumption?

(b) How much energy would be consumed in the optimal case of the previous item?

(c) Assume now an energy model which determines the energy required to send a message from A to B as $E(A, B) = d(A, B)^\alpha + c$, with $c > 0$. Argue why this energy model is more realistic.

(d) Prove under the modified energy model introduced in previous item that there exists an optimal number n of equidistant intermediate nodes between S and D that minimizes the overall energy consumption when using these intermediate nodes in order to route a packet from S to T. [Assume n as a continuous variable for simplicity].

(e) Derive a closed-form expression on how much energy will be consumed when using this optimal number n of relay nodes. [Assume n as a continuous variable for simplicity].

Problem 2.3 Density of a Function of a Random Variable: the Rayleigh channel
Suppose that $\mathbf{x}$ has a chi-square distribution with the density

$$f(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2} U(x),$$

where

$$\Gamma(a+1) = \int_0^\infty x^a e^{-x} dx$$

is the gamma function and $U(x) = 1$ for $x \geq 0$ and $U(x) = 0$ otherwise. For a new random variable $\mathbf{y} = \sqrt{\mathbf{x}}$ compute its density function.

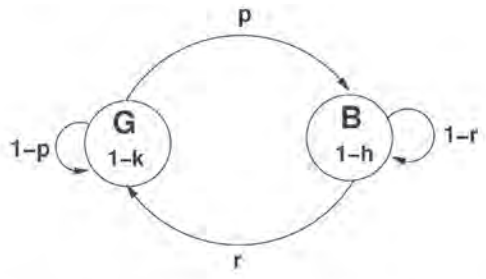


Figure 2.7 2-state Markov chain describing to Gilbert Elliott model.

Problem 2.4 Deriving the Density of a Function of a Random Variable: The step windowing

For a random variable $\mathbf{x}$ with density function f_x, compute the density function of $\mathbf{y} = \mathbf{x}U(\mathbf{x})$, where

$$U(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} .$$

Problem 2.5 Deriving the Density of a Function of a Random Variable: The shadow fading

A log-normal distribution is a continuous probability distribution of a random variable whose logarithm has a Normal distribution. If $\mathbf{x}$ is a random variable with a normal distribution, then $\mathbf{y} = \exp(\mathbf{x})$ has a log-normal distribution. For $\mathbf{x} \sim \mathcal{N}(\mu, \sigma)$, compute the density function of $\mathbf{y} = \exp(\mathbf{x})$.

Problem 2.6 Mean and Variance of Log-normal Distribution

For $\mathbf{x} \sim \mathcal{N}(\mu, \sigma)$, compute mean and variance of $\mathbf{y} = \exp(\mathbf{x})$.

Problem 2.7 Gillbert-Elliott Model for Wireless Channels

The Gillbert-Elliott model is a 2-state Markov chain to model the wireless channel behavior when sending packet losses. This model consists of two channel states denoted as Good and Bad with corresponding error probabilities. In Fig. 2.7 each state may introduce errors for independent events with state dependent error rates $1 - k$ in the good and $1 - h$ in the bad state. In our framework, we interpret the event as the arrival of a packet and an error as a packet loss.

(a) Based on the given error rates and transition probabilities p and r, formulate π_G and π_B to be the stationary state probabilities of being in each state.

(b) Obtain error rate p_E in stationary state.

(c) Consider the Average Error Length (AEL) and Average number of Packet Drops (APD) as two statistics of channel. Derive π_G and π_B.

Problem 2.8 Gillbert-Elliott model application

We have two sensor nodes that share a wireless channel. The state of the channel follows the Gillbert-Elliott model. Suppose that the transition probabilities in Fig. 2.7 are $p = 10^{-5}$ and $r = 10^{-1}$.

(a) Find the average length of an error burst.

(b) Obtain the average length of an error-free sequence of message transmission.

(c) Assume that the error probability in Good and Bad states is negligible and almost sure, respectively. Compute the average message loss rate of the channel.

Chapter 3

Physical Layer

The nodes of a WSN have to be able to successfully transmit/receive messages over the wireless channel that serves as physical medium for the digital communication link between the nodes. At the physical layer of the protocol stack, a reliable communication depends upon radio power, wireless channel attenuation, modulation and coding. The aim of this chapter is to understand the basics of physical layer so to model mathematically the probability to successfully receive messages as function of the radio power, modulations, coding, and channel attenuations normally experienced in WSNs.

In WSNs commercially available, the physical layer uses three different bands (also often called ranges) of frequencies for communication. In practice, a frequency slot over a frequency band is used to transmit message. The frequency slot corresponds to the carrier frequency f_c that is used (see previous chapter). Such a frequency slot is often called "frequency channel" or communication channel. The three frequency bands for home and industrial automation purposes of WSNs are 2.4GHz, 915MHz in America, and 868MHz in Europe. In addition to larger bands reserved for Ultra Wide Bands (UWB) communication (which we do not consider in this book), 47 frequency channels are distributed among these bands. To have an idea of the about of bits per seconds that can be transmitted over these frequency channels, we mention that one channel is associated with the 868MHz band with a data rate of 20–250 kbps; 30 channels are defined for the 915MHz band and 16 are used in the 2.4GHz range. Over these frequencies, and using the typical transmit radio powers of the transceivers of commercially available circuits, the transmission range of the nodes results to be 10–100 m with data rates from 20 to 250 kbps. In the 868/915MHz bands, the modulation used over the carrier frequencies are Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK). QPSK is also used in the 2.4GHz band. This is the reason why in this chapter, we give particular relevance to these modulation formats.

The chapter begins with discussion on basic components of digital com-

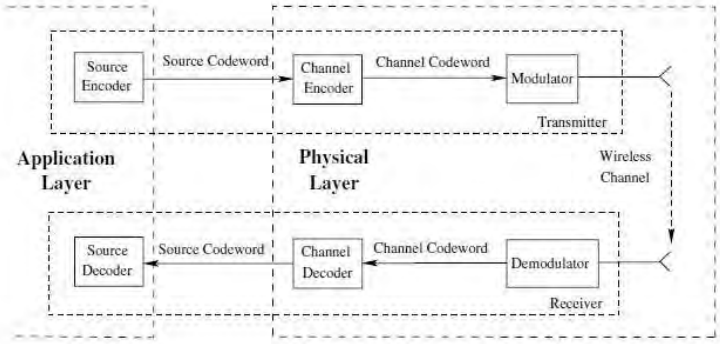


Figure 3.1 Overview of radio frequency (RF) communication blocks.

munication system which is followed by different types of digital modulation techniques. Then communication over AWGN and fading channels is discussed. The error probability calculations for both the aforementioned cases of channels is derived. At the end, a quick mention to channel coding theory is given with particular focus on the block codes.

3.1 Basic Components

Low-power RF communication techniques are generally used in WSNs. An illustration of RF wireless communication and general components is given in Figure 3.1. Accordingly, the following are performed to transmit information between a transmitter and a receiver:

- Source coding (data compression): At the transmitter end, the information source is first encoded with a message source encoder, which exploits the properties of the source message to encode it in a number of bits, and produce source message codeword. Thus, a source message codeword is a sequence of bits corresponding to the source message. Source coding is also referred to as data compression, because the goal is to encode a source message into a few bits as possible. Source (en/de)coding is performed at the application layer as shown in Figure 3.1.

- Channel coding (error control coding): The source codeword is then further encoded by the channel encoder. The purpose of this additional encoding is to add robustness to the source codeword with respect to the wireless channel errors affecting the transmitted bits. Therefore, channel coding is also referred to as error control coding.

- Interleaving and modulation: The bits that result from the channel coding are then interleaved to combat the bursty errors that can affect

a large number of consecutive bits. The channel coding and the interleaving mechanism help the receiver either to (1) identify bit errors to initiate retransmission or (2) correct a limited number of bits in case of errors. The bits are then associated to an analog signal by the modulation procedure. In this operation, the bits resulting from the channel coding are grouped into groups each of which is associated to a "message symbol". In particular, an analog signal (or a set thereof) is modulated by the message symbol (and hence by the sequence of bits the symbol carries) to create the waveform that will be sent over the wireless channel. Finally, the waveforms are transmitted through the antenna to the receiver.

- Wireless channel propagation: The transmitted waveform, which is essentially an electromagnetic wave, travels through the wireless channel. As a consequence, the waveform is attenuated and distorted by several wireless channel effects.

At the receiver end, symbol detection is performed first to lock into the sent waveform. The received signal is demodulated to extract the message symbols, which are then de-interleaved and decoded by the channel and the source decoders to determine the information sent by the transmitter.

3.2 Modulation

After encoding the source message by source coding and channel coding, the resulting bits are divided into groups. Every groups is then associated to a message symbol, which is then modulated at the transmitter. The modulated signal is then transmitted over the wireless channel by a node's antenna. Such a signal has the following representation:

$$s\left(t\right)=\sum_{k=0}^{\infty}a_{k}\left(t\right)g\left(t-kT_{s}\right), \tag{3.1}$$

where a_k is complex valued function representing the modulated message symbol and $g(t)$ is pulse shaping function of that modulated symbol. T_s is the symbol duration.

It is interesting to compute the Fourier transform of $a_k(t)g(t-kT_s)$ because it helps to understand how the modulated signal occupies frequency bands around the carrier frequency. The Fourier transform is given by

$$G\left(f\right)\triangleq\int_{0}^{T_{s}}a_{0}g\left(t\right)e^{-2\pi ft}dt, \tag{3.2}$$

and power spectral density of the signal is

$$\Phi_s(f) \triangleq \frac{|G(f)|^2}{T_s}. \tag{3.3}$$

The power spectral density gives a measure of occupation of the modulated signal in the frequencies around the carrier frequency.

The modulated message symbols, a_k, can be modulated by different techniques. There are many types digital modulation techniques such as ASK (Amplitude Shift Keying), PSK (Phase Shift Keying), FSK (Frequency Shift Keying), QAM (Quadrature Amplitude Modulation) and others. The techniques that are relevant for WSNs commercially available are discussed in next sections.

3.2.1 Binary Phase Shift Keying (BPSK)

A basic digital modulation technique is BPSK (also sometimes called PRK, Phase Reversal Keying, or 2PSK) which is the simplest form of phase shift keying (PSK).

The general form for BPSK follows the equation:

$$s_n(t) = \sqrt{\frac{2E}{T_s}}\cos(2\pi f_c t + \pi(1-n)) \qquad n = 0, 1. \tag{3.4}$$

In this case, $a_k(t)$ and $g(t)$ in Equation (3.1) becomes

$$a_k(t) = \begin{cases} \cos(2\pi f_c t) & \text{if bit 0 is transmitted} \\ \cos(2\pi f_c t + \pi) & \text{if bit 1 is transmitted} \end{cases} \tag{3.5}$$

$$g(t) = \sqrt{\frac{E}{T_s}}, \; 0 \le t \le T_s. \tag{3.6}$$

In frequency domain, $a_i(t)g(t)$ is

$$\begin{aligned} G(f) &= -\sqrt{\frac{E}{T_s}}\frac{e^{j\pi f T_s} - e^{-j\pi f T_s}}{j2\pi f} = -\sqrt{\frac{E}{T_s}}T_s\frac{\sin(\pi f T_s)}{\pi f T_s} \\ &= -\sqrt{\frac{E}{T_s}}T_s \mathrm{sinc}(f T_s), \end{aligned} \tag{3.7}$$

and spectral density of BPSK becomes:

$$\Phi_s(f) \triangleq \frac{E}{T_s}\mathrm{sinc}^2(f T_s), \tag{3.8}$$

where we see that the transmit power is

$$P_t = \frac{E}{T_s}.$$

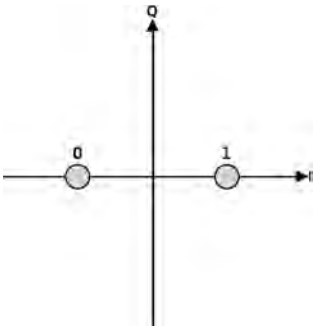


Figure 3.2 Constellation of a BPSK modulation.

If we look at Equation (3.5), we see that it uses two phases which are separated by 180° on the constellation diagram, so it can also be termed 2-PSK. The placement of constellation points is not fixed. For example, in Figure 3.2, they are shown on the real axis, at 0° and 180°.

This modulation is the most robust of all the PSKs since it takes the highest level of noise or distortion to make the demodulator reach an incorrect decision. It is, however, only able to modulate at 1 bit/symbol (Figure 3.2) and so is unsuitable for high data-rate applications. In the presence of an arbitrary phase-shift introduced by the communications channel, the demodulator is unable to identify the modulation points correctly. As a result, the data is often differentially encoded prior to modulation. BPSK is functionally equivalent to 2-QAM modulation.

3.2.2 Quadrature Phase Shift Keying (QPSK)

QPSK uses a constellation signal with four phases that are separated from each other by 90°. It can modulate two bits per symbol and therefore can achieve higher data rate than BPSK, however the probability of error increases as well. The constellation diagram for QPSK is shown in Figure 3.3.

The general form for BPSK follows the equation

$$s_n(t) = \sqrt{\frac{2E_s}{T_s}} \cos\left(2\pi f_c t + \frac{\pi}{4}(2n-1)\right), \; n = 1, 2, 3, 4\,. \tag{3.9}$$

The QPSK modulation scheme has been adopted by the IEEE 802.15.4 standard for WSNs (which we will study in the detail in next chapter) to modulate the chips (bits used to represents an information bit after the spread spectrum coding) sent for each bit as a part of the direct sequence spread spectrum (DSSS) scheme. More specifically, modulation of bits in an IEEE 802.15.4 transceiver uses offset QPSK with DSSS. The modulation structure consists of the conversion of bits to symbols, conversion of symbols

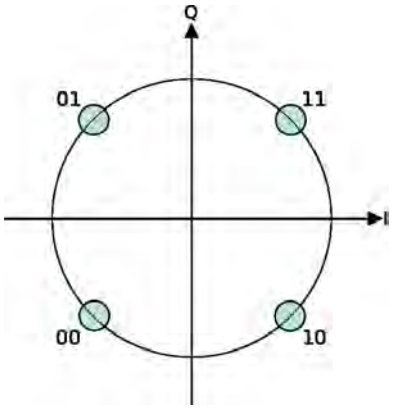


Figure 3.3 QPSK constellation diagram.

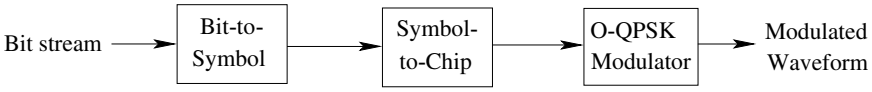


Figure 3.4 Modulator structure of the IEEE 802.15.4 standard for WSNs.

to chips, and QPSK modulation for these chips. The modulation structure is shown in Figure 3.4.

The modulation scheme used in Telos nodes is O-QPSK with DSSS. The bit error rate of this scheme is given by:

$$\mathrm{P}_{b,\mathrm{QPSK}} = Q\left(\sqrt{\frac{E_c}{N_0}}\right), \tag{3.10}$$

with

$$\frac{E_c}{N_0} = \frac{2N\frac{E_b}{N_0}}{N + \frac{4}{3}\frac{E_b}{N_0}(K-1)}, \tag{3.11}$$

where N is the number of chips per bit, and K is the number of simultaneously transmitting users.

3.2.3 Amplitude Shift Keying

Amplitude Shift Keying (ASK) is a digital modulation technique in which the amplitude of an analog carrier singal is modulated according to the message bits. The frequency and phase of the carrier signal remain unchanged. ASK can be realized by different schemes. A basic one is called on-off keying. In this approach, the bit '1' can be represented as presence of the signal having frequency f_c while the bit '0' can be represented by no signal. This

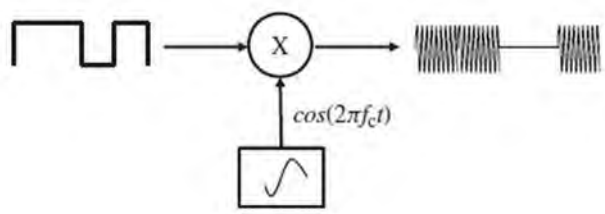


Figure 3.5 ASK using on-off switch

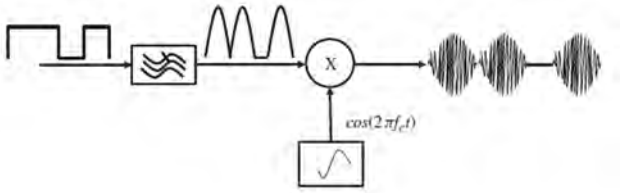


Figure 3.6 ASK modulation using pulse shaping filter.

is illsutrated in Figure 3.5. It can be seen that the mixer multiplies the message and carrier signal(from the local oscillator) to produce the modulated wave.

Direct mixing of square wave requires mixer with excessive bandwidth, which is expensive to afford. This can be avoided by using pulse-shaping filter (PSF) that is implemented before the mixer. PSF can remove high frequency components from the square wave and convert it into low-frequency signal that will modulate the carrier signal. This is shown in Figure 3.6.

On-off keying is an example of binary-ASK. More amplitude values can be added for large signal sets, for example 4-ASK,8-ASK etc. Every point (symbol) is associated to a signal with specific amplitude. In this case, plural bits constitute a symbol of the signal set that results in large data rate but at the same time, high error probability. In ASK, all the constellation points reside on one axis and distance between these points determines the error probability. For fixed peak power, plural bits mean less distance between the constallation points and hence high probability of error.

3.3 Communication over Gaussian Channel

Signals in the Gaussian channel are received corrupted by Additive White Gaussian Noise (AWGN). We assume that such as noise has a power spectral density of $N_0/2$. Signal and noise are orthogonal with each other so receivers use matched filters and correlators to separate out the two components. In particular, at the receiver, after the matched filter, the signal is mapped onto the signal space of the modulation scheme. A decision is then takes as to

which symbol correspond to the point on the constellation. These decisions are made by the decision circuit based upon the closest points in signal space.

While designing signal sets, the minimum distance between the points in signal space is maximized and it is also subject to peak and average power constraints. This is because of the requirement for low error rates in digital communications. The symbol error probability is closely approximated as

$$\mathrm{P}(e) \simeq N_{d\,\min} Q\left(\frac{d_{\min}}{\sqrt{2N_0}}\right), \tag{3.12}$$

where $N_{d\,\min}$ is the average number of nearest neighbors at the minimum distance.

3.3.1 Error Probability for BPSK

Assume that the transmitted signal is received together with an Additive White Gaussian Noise (AWGN)

$$r(t) = s(t) + n_0(t), \tag{3.13}$$

$$n_0(t) \in N\left(0, \sigma^2 = \frac{N_0}{2T_s}\right). \tag{3.14}$$

After the matched filter, the demodulator in the receiver block produces a signal

$$r'(t) = s'(t) + n_0'(t), \tag{3.15}$$

$$n_0'(t) \in N\left(0, \sigma^2 = \frac{N_0}{2T_s}\right). \tag{3.16}$$

Such a demodulated signal contains both the message signal and noise. Here we have that

$$s'(t) = \pm\sqrt{\frac{E}{T_s}},$$

so the bit '0' can be chosen as the threshold between the two points of the signal space. If $r'(t) \geq 0$ the detector decides for bit 1 and if $r'(t) < 0$ the detector decides for bit 0. The received noise has zero mean and variance $N_0/2T_s$. An error occurs when the decision circuits decides '1' while '0' was transmitted and vice versa. Calculating conditional probabilities for BPSK gives

$$\mathrm{P}_{e,0|1} = \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x-\sqrt{\frac{E}{T_s}}^2}{2\sigma^2}} dx = Q\left(\sqrt{\frac{2E}{N_0}}\right), \tag{3.17}$$

$$\mathrm{P}_{e,1|0} = \int_{0}^{-\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x+\sqrt{\frac{E}{T_s}}^2}{2\sigma^2}} dx = Q\left(\sqrt{\frac{2E}{N_0}}\right). \tag{3.18}$$

The overall probability of error is given by

$$\mathrm{P}_e = \frac{1}{2}\mathrm{P}_{e,0|1} + \frac{1}{2}\mathrm{P}_{e,1|0} = Q\left(\sqrt{\frac{2E}{N_0}}\right), \tag{3.19}$$

where the Signal to Noise Ratio is defined as

$$\mathrm{SNR} \triangleq \frac{E}{N_0}. \tag{3.20}$$

3.3.2 Error Probability for 4-PAM

Pulse amplitude modulation (PAM) is effected by multiplying a rectangular pulse of duration T_s, the symbol time, by one of M equally spaced voltage levels symmetric about the origin. It can be easily shown that this symmetry minimizes the peak and average power, without affecting the error probability. In the particular case of 4-PAM, the signals are given by

$$\begin{aligned} s_{00} &= \sqrt{E/5}\phi_1(t) \qquad & s_{01} &= 3\sqrt{E/5}\phi_1(t) \\ s_{10} &= -\sqrt{E/5}\phi_1(t) \qquad & s_{11} &= -3\sqrt{E/5}\phi_1(t) \end{aligned} \tag{3.21}$$

with

$$\phi_1(t) = \sqrt{\frac{1}{T_s}}.$$

Clearly, the average energy is $E = (2E/5 + 9E/5)$, and the squared minimum distance is 4E/5. Here s_{00} and s_{10} have two neighbors at the minimum distance, while s_{11} and s_{01} have only one. Thus $N_{d\,\min} = 1.5$. Also, the most likely error is to cross only into the neighboring region, for which the bit labels differ in only one position. Thus a symbol error results in only one of the two bits being in error. Consequently, symbol error rate:

$$\mathrm{P}_e \simeq 1.5Q\left(\sqrt{\frac{2E}{5N_0}}\right) = 1.5Q\left(\sqrt{\frac{4E_b}{2N_0}}\right), \tag{3.22}$$

with bit error rate

$$\mathrm{P}_b \simeq 0.75Q\left(\sqrt{\frac{4E_b}{5N_0}}\right),$$

and the energy per bit is

$$E_b = \frac{E}{\log_2 M} = \frac{E}{2}.$$

Modulations may be compared on the basis of their error performance as a function of E_b/N_0, and their spectral efficiencies. Binary PAM for example has two signals: $s_0 = \sqrt{E}\phi_1(t)$, $s_1 = -\sqrt{E}\phi_1(t)$.

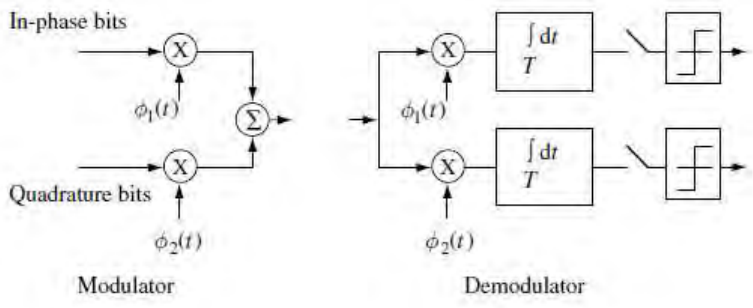


Figure 3.7 Modulator and demodulator for 4-QAM.

Given that the same pulse shaping function is employed, the psd is identical to that of binary PSK and M-ary PAM, for any M. Compared to 4-PAM, only half as many bits per hertz may be conveyed. It is therefore less spectrally efficient. The error probability is given by:

$$\mathrm{P}_e = \mathrm{P}_b = Q\left(\sqrt{\frac{2E}{N_0}}\right) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) . \tag{3.23}$$

Thus, binary PAM requires less energy per bit to achieve the same error probability as 4-PAM, and is considered more energy-efficient. The ratio is 5/2 or approximately 4 dB.

3.3.3 Error Probability for QAM

4-QAM and 4-PSK are actually the same modulation. When described by four equally spaced phases:

$$s_i(t) = \sqrt{\frac{2E}{T}}\cos(2\pi f_c t + (2i-1)\pi/4), \qquad 0 \le t \le T\,, \tag{3.24}$$

it is 4-PSK, whereas it is thought of as 4-QAM when equivalently described as 2-PAM on quadrature carriers:

$$s_i(t) = \sqrt{\frac{2E}{T}}\cos((2i-1)\pi/4)\cos(2\pi f_c t) - \sqrt{\frac{2E}{T}}\sin((2i-1)\pi/4)\sin(2\pi f_c t)\,. \tag{3.25}$$

In either case, the signal space can be constructed using only two basis functions:

$$\begin{aligned} \phi_1(t) &= \sqrt{\frac{2}{T}}\cos(2\pi f_c t)\,, \qquad 0 \le t \le T \\ \phi_2(t) &= \sqrt{\frac{2}{T}}\sin(2\pi f_c t)\,, \qquad 0 \le t \le T \end{aligned} \tag{3.26}$$

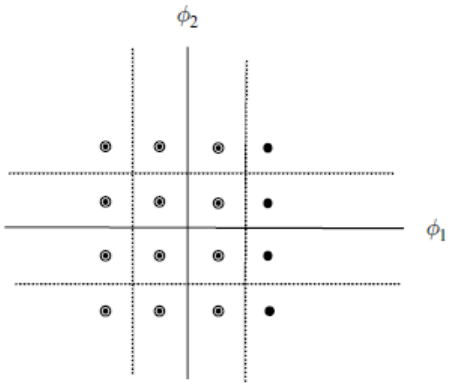


Figure 3.8 Decision regions for 16-QAM.

The minimum distance is $\sqrt{2E}$, and there are two neighbors at $d_{\min}$. Consequently:

$$\mathrm{P}_e \simeq 2Q\left(\sqrt{\frac{E_0}{N_0}}\right) = Q\left(\sqrt{\frac{2E_b}{2N_0}}\right) . \tag{3.27}$$

This is as efficient in terms of E_b/N_0 as BPSK or PAM, with twice the bits per symbol. It uses the same bandwidth, and is therefore twice as spectrally efficient. The modulator and demodulator are shown in Figure 3.7. In the modulator, the input streams of 1s and 0s are converted to 1s and –1s. In the demodulator, the integrator outputs are sampled every T_s seconds, and then a threshold detector decides the value is 1 if the result is positive or 0 otherwise. M-QAM generally uses larger PAM constellations on each of the quadrature carriers. For 16-QAM, draw the optimal decision regions in signal space and a receiver that would implement these decisions, then compute the approximate error probability.

Solution

There remain only two orthogonal functions. The decision boundaries are composed of the right bisectors of the lines joining neighbors. Due to the symmetry of the signal set, the boundaries are as depicted in the figure below:

The demodulator is exactly as depicted in Figure 3.7, except that four-level uniform slicers replace the two-level slicers (quantizers). Let the coordinates of the lowest-energy signal in the first quadrant be (a, a). Then the minimum distance is 2a, and the average number of neighbors at this distance is $[4(4) + 8(3) + 4(2)]/16 = 3$. The average energy E may be computed using the four-fold symmetry and similarity for the two coordinates as $[4(a)^2 + 4(3a)^2]/4 = 10a^2$. Alternatively one could have doubled the average

energy of 4-PAM. In either case, $d_{\min}$ is related to E_b by

$$d_{\min} = 2a = 2\sqrt{\frac{E}{10}} = 2\sqrt{\frac{4E_b}{10}} = \sqrt{\frac{8E_b}{5}}\,. \tag{3.28}$$

So that

$$\mathrm{P}_e = 3Q\left(\sqrt{\frac{4E_b}{5N_0}}\right)\,.$$

Gray coding, which labels symbols so that the nearest symbols differ in exactly one bit, exists for all QAM configurations, and thus the bit error rate performance as a function of E_b/N_0 is the same for 16-QAM as for 4-PAM. This is achieved with twice the spectral efficiency.

Signal sets can also be constructed using orthogonal frequencies. In BFSK two carriers separated in frequency by 1/T are used, with the bit conveyed depending upon the frequency. A two-branch detector is required. The squared distance between two orthogonal signals with energy E is half that of two antipodal signals with energy E, and thus BFSK performs 3 dB worse with respect to E_b/N_0 than BPSK. In contrast to other M-ary modulations, however, M-FSK actually has better performance as M increases, with

$$\mathrm{P}_e \simeq (M-1)Q\left(\sqrt{\frac{E}{N_0}}\right) = (M-1)Q\left(\sqrt{\frac{E_b \log_2 M}{N_0}}\right)\,. \tag{3.29}$$

The price of this performance increase is a linear increase in the required bandwidth, in contrast to the phase/amplitude modulations for which bandwidth is independent of M. In practice, a better trade in bandwidth and E_b/N_0 performance is obtained through the use of channel codes and a modulation such as PSK, making these the methods of choice for deep-space communication. FSK finds application mainly as a non-coherent scheme in low-cost applications such as paging.

3.4 Communication over Fading Channel

Signal propagation in real channel is different from that in Gaussian channel. It not only suffers from AWGN but also the transmitted signals are attenuated and distorted by the wireless channel.

$$r\left(t\right) = \sqrt{A}s\left(t\right) + n_0\left(t\right)\,. \tag{3.30}$$

Electromagnetic waves in the air suffers from path loss, fast fading or slow fading. The fading effects of wireless medium are random so they are modelled by probability distributions. For details, see Chapter 2. The received

power in presence of AWGN + Rayleigh channel with path loss, Rayleigh fast fading, and fixed shadow fading can be modelled as,

$$P_r = P_t G_t\left(\theta_t, \psi_t\right) G_r\left(\theta_r, \psi_r\right) \frac{\lambda^2}{(4\pi r)^2} \overline{\mathrm{PL}} \cdot y \cdot z \triangleq P_t C \cdot z \tag{3.31}$$

where

$$C = G_t\left(\theta_t, \psi_t\right) G_r\left(\theta_r, \psi_r\right) \frac{\lambda^2}{(4\pi r)^2} \overline{\mathrm{PL}} \cdot y \tag{3.32}$$

The receiver sees the transmit power $P_t = E/T_s$ received with an attenuation $C \cdot z$ and then

$$\mathrm{SNR} = \frac{E}{N_0} C z\,. \tag{3.33}$$

To compute the probability, mathematically, it is as if the transmit power were $P_t = E/T_s C z$ and the channel were just AWGN and with no fading. So, the probability of error with fading has same expression as that of simple AWGN channels, but with the new SNR as defined above. For example

$$\mathrm{P}_{e,\mathrm{BPSK}} = Q\left(\sqrt{\frac{2E}{N_0}}\right) \longrightarrow \mathrm{P}_{e,\mathrm{BPSK}}(z) = Q\left(\sqrt{\frac{2ECz}{N_0}}\right), \tag{3.34}$$

$$\text{AWGN} \quad \longrightarrow \quad \text{AWGN + Rayleigh fading}\ .$$

We have derived instantaneous error probability over fading channel so far which depends on the given realization of the fading channel z. In order to calculate the average probability of error, take the expectation of $\mathrm{P}_{e,\mathrm{BPSK}}(z)$ over the distribution of z, so

$$p(z) = \frac{1}{\gamma^*} e^{-\frac{z}{\gamma^*}}, \qquad \gamma* = \mathrm{SNR} = \frac{E}{N_0} C, \ \ z = 1\,, \tag{3.35}$$

$$\mathrm{P}_{e,\mathrm{BPSK}} = \int_0^\infty P_{e,\mathrm{BPSK}}(z) p(z) dz = \frac{1}{2}\left[1 - \sqrt{\frac{\gamma^*}{1+\gamma^*}}\right]. \tag{3.36}$$

At high SNR, the approximation $(1+x)^{\frac{1}{2}} = 1 + x/2$ can be used, giving

$$\mathrm{P}_e \simeq \frac{1}{4\gamma^*} \tag{3.37}$$

compared with $\mathrm{P}_e = Q(\sqrt{2\gamma^*})$ for the Gaussian channel.

Thus for the Gaussian channel a linear increase in SNR results in an exponential decrease in the error probability,

$$\mathrm{P}_{e,\mathrm{BPSK}} = Q\left(\sqrt{\frac{2E}{N_0}}\right), \tag{3.38}$$

but incase of Rayleigh channel only a linear decrease in the error probability occurs on linear increase of the SNR. This is disastrous: to achieve $P_e = 10^{-6}$, the Rayleigh channel demands 40dB higher SNR than the Gaussian channel. As this represents a factor of 10000 in linear scale, it is clear that small low-power devices will not be adequate if boosting power is the only approach.

$$P_{e,\text{BPSK}} \simeq \frac{1}{4\gamma^*}, \qquad \gamma^* = \text{SNR} = \frac{E}{N_0}C. \tag{3.39}$$

There is nothing special about BPSK in this respect; in fact the behavior is typical. For the popular binary modulations the error probabilities are given approximately by:

$$\begin{aligned} &\text{coherent PSK} = 1/4\gamma^*, \qquad &&\text{differential PSK} = 1/2\gamma^*, \\ &\text{coherent FSK} = 1/2\gamma^*, \qquad &&\text{noncoherent FSK} = 1/\gamma^*. \end{aligned} \tag{3.40}$$

Incase of Gaussian channel, both coherent and non-coherent modulations show similar performances at high SNR in P_e vs.E_b/N_0. But for the Rayleigh channel, low SNR behaviour of being 3dB worse is always observed. This is due to the deep fade events at low SNR that causes most of the errors in Rayleigh channel. Thus receiver has to suffer more for lack of coherence.

There are two basic approaches to dealing with this problem (three if one counts simply giving the customer a low quality of service). The first is to allocate more bits to periods of good SNR, which is possible if the channel is only slowly fading and the transmitter and receiver can collaborate. The optimal allocation is similar to that in OFDM, only applied to the time domain. This may be done either to maximize the transmission rate, or to achieve a desired transmission rate with the minimum required energy. The second is to spread the information over the fading events, with application of appropriate weighting as the signals are recombined to make the channel seen by an information bit more like the average condition. This second approach amounts to diversity combining. An aggressive combination of the approaches can completely overcome multipath fading if the channel can be estimated accurately.

In fact, if transmission delay is not an issue and the number of data to send is small, multipath fading can even be advantageous. A low-energy signal is used to probe the channel. The far-end user then signals when the SNR is far above γ^* and a low-power signal can then be used to convey a large number of bits per symbol. This game can, of course, also be played in the frequency domain if the fading is frequency-selective.

3.5 Channel Coding (Error Control Coding)

Source coding techniques aim to decrease the amount of data to be transmitted by exploiting the statistical properties of the information. Once encoded, this information needs to be transmitted reliably over the wireless

channel. However, the transmitted information can be easily corrupted due to the adverse effects of the wireless channel. To combat these effects, channel coding schemes have long been investigated in the context of wireless communication theory. The main goal of these channel coding approaches is to exploit the statistical properties of the channel to inject redundancy into the information to be sent. Consequently, the received information can be decoded successfully even if certain portions of it are distorted. Channel coding is also referred to as error control coding (ECC) or forward error correction (FEC).

When a certain number of information bits are provided to the source encoder, an output that consists of a smaller number of bits is created. Although compressed, if this source codeword is successfully received at the source decoder, the original set of information bits can be decoded. In order to successfully transmit the source codeword, the channel encoder creates an output that consists of a larger number of bits compared to the source codeword. The redundant bits (or parity bits) are added to the source codeword to create the channel codeword, which helps combat the wireless channel errors.

There exist several powerful channel codes that have been developed for communication systems. These codes constitute the basis of error control techniques such as automatic repeat request (ARQ), forward error correction (FEC), and hybrid ARQ schemes. Next, we present some of the common channel codes that are being used in WSNs.

3.5.1 Block Codes

Block codes are generally preferred in WSNs due to their relatively simpler implementation and smaller memory requirements. A block code transforms an input message u of k bits into an output message v of n bits, $n > k$. The output message v is denoted as the channel codeword. Depending on the complexity of the code, each code is capable of correcting up to t bits in error. Accordingly, a block code is identified by a tuple (n, k, t).

The error detection and error correction capabilities of block codes are determined by the minimum distance of the code. The Hamming distance between two codewords is defined as the number of places they differ. Accordingly, the minimum Hamming distance, dmin, is the minimum distance between any two words in a code. A code with minimum distance dmin can detect up to $dmin - 1$ errors and correct up to t errors such that $2t + 1 \leq dmin \leq 2t + 2$.

Three main types of block codes are used in WSNs in general.

BCH Codes

A popular example of block codes is the Bose, Chaudhuri, and Hocquenghem (BCH) code. BCH codes have been used in many different applications and provide both error detection and correction capabilities. A BCH code is identified by a tuple (n, k, t), where n is the block length, k is the information length, and t is the maximum number of errors that can be corrected. The following hold true for any BCH code:

$$n = 2^m - 1\,, \tag{3.41}$$

$$n - k \leq mt\,, \tag{3.42}$$

$$d_{\min} \geq 2t + 1\,. \tag{3.43}$$

The encoding and decoding operations of BCH codes are performed in a finite field GF(2^m), called a Galois field, which has 2^m elements.

Reed Solomon (RS) Codes

RS (Reed–Solomon) codes are a family of BCH codes that are non-binary, i.e., the operations are performed in GF(q), where q is a prime number. While they retain the properties of binary BCH codes, the following hold true for all RS codes:

$$n = q - 1\,, \tag{3.44}$$

$$n - k = 2t\,, \tag{3.45}$$

$$d_{\min} = 2t + 1\,. \tag{3.46}$$

Cyclic Redundancy Check (CRC) Codes

CRC (Cyclic Redundancy Check) codes are a special family of BCH codes that are used to detect errors in a packet. Irrespective of whether an error control code is used within communication, CRC codes are used in almost any communication system. The automatic repeat request (ARQ) scheme, which relies on retransmissions for reliability, is based on CRC codes. In particular, CRC codes are BCH codes with $d_{min} = 4$. Upon decoding, CRC codes detect whether a packet is received in error or not. However, they cannot correct these errors. Block codes are easy to implement because of the relatively simpler encoder and decoder structures. Consequently, the encoding complexity of block codes is negligible. Hence, only the decoding complexity is considered. Accordingly, assuming software implementation, block codes can be decoded with $(2nt + 2t^2)$ additions and $(2nt + 2t^2)$ multiplications.

Problems

Problem 3.1 Bit error probability for BPSK over AWGN channels

Compute the probability of error for binary phase shift keying (**BPSK**) with Additive white Gaussian noise (AWGN) channel model.

Problem 3.2 Bit error probability for QPSK over AWGN channels

Compute the probability of error for Quadrature phase-shift keying (**QPSK**) modulation with Additive white Gaussian noise (AWGN) channel model.

Problem 3.3 Error probability for 4-PAM over AWGN channels

Compute the probability of error for Pulse amplitude modulation (PAM) with Additive white Gaussian noise (AWGN) channel model.

Problem 3.4 Average error probability for Rayleigh fading

Compute the average probability of error for a Rayleigh fading channel given the error probability of AWGN channel model.

Problem 3.5 Detection in a Rayleigh fading channel

In a Rayleigh fading channel the detection of symbol x from y is based on the sign of the real sufficient statistic

$$r = |h|x + z,$$

where $z \sim \mathcal{N}(0, N_0/2)$. It means that, If the transmitted symbol is $x = \pm a$, then, for a given value of h, the error probability of detecting x is

$$\mathbf{Q}\left(\frac{a|h|}{\sqrt{N_0/2}}\right) = \mathbf{Q}\left(\sqrt{2|h|^2\mathrm{SNR}}\right),$$

where $\mathrm{SNR} = a^2/N_0$ is the average received signal-to-noise ratio per symbol time (note that we normalized the channel gain such that $\mathbf{E}[|\mathbf{h}|^{\mathbf{2}}] = \mathbf{1}$.) For Rayleigh fading when $|h|$ has Rayleigh distribution with mean 0 and variance 1, calculate the average probability of error. Approximate the solution for high SNR regions.

Problem 3.6 Average error probability for log-normal fading

Consider a log-normal wireless channel with AWGN receiver noise. We know that the probability of error in AWGN is

$$\mathbf{Q}(\gamma) = \Pr\{\mathbf{x} > \gamma\} = \int_{\gamma}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \,.$$

The average probability of error with respect to the log-normal distribution is the average of $\mathbf{Q}(\gamma)$ with respect to the log-normal distribution. It is difficult to compute because $\mathbf{Q}$ is highly non linear. Suppose to perform a Stirling approximation of the $\mathbf{Q}$ function, which is

$$\mathbb{E}\{f(\theta)\} \sim \frac{2}{3} f(\mu) + \frac{1}{6} f(\mu + \sqrt{3}\sigma) + \frac{1}{6} f(\mu - \sqrt{3}\sigma) \,.$$

where $f(\theta)$ is any function of a random variable θ having mean μ and variance σ^2. Compute the average probability of error of log-normal channel by using the Stirling approximation.

Problem 3.7 Probability of error at the message level

In a WSN communication platform, consider a Rayleigh Channel over a AWGN receiver noise. The message is a frame of size f bits and is composed of the preamble, network payload, and a CRC code.

(a) Compute p the probability that the message is correctly received.

(b) Assume that the received signal level at the receiver decays inversely with the squared of the distance, i.e.,

$$\text{SNR} \approx \frac{\alpha E_b}{N_0 d^2} \,.$$

For messages of size 10 bits and the values $E_b/N_0 = 100$ and $\alpha = 0.1$, compute the farthest distance to deploy a receiver such that the probability of successfully message reception is at least $p = 0.9^{10} \approx 0.35$.

Problem 3.8 Gray Code (Ex. 6.5 in (Pottie and Kaiser, 2005))

The property of the Gray code is that the code-words of adjacent symbols only differ in one bit. For example, the code words of 8-**PAM** (pulse amplitude modulation) symbols are as illustrated in Figure 3.9. This results in a minimum expected number of bit errors per symbol error, in conditions of low symbol error probability. Devise a Gray code for 16-**QAM** (quadrature amplitude modulation) symbols.

Problem 3.9 The rate 2/3 parity check code

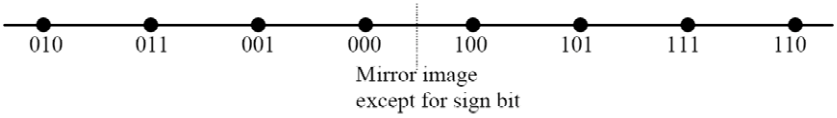


Figure 3.9 Gray-coded 8-PAM.

A parity check code forms a modulo 2 sum of information bits and then appends this bit as the parity check. Consider, e.g., a scheme in which there are two information bits and one party bit. The codewords are then the set $000, 011, 101$, and 101, which have even parity, while the odd-parity possibilities $001, 010, 100$ and 111 are excluded. There are four code-words and thus two bits of information, compared with the eight uncoded possibilities which would take three bits to represent. The code rate is thus $2/3$. Suppose this coding scheme is used in conjunction with binary phase shift keying (**BPSK**). Compute the coding gain assuming **ML** soft coding.

Problem 3.10 Hard vs. soft decisions (Ex. 6.7 in (Pottie and Kaiser, 2005))

In previous exercise, if hard-decisions are used instead of soft-decisions, answer to the following questions:

(a) How many errors can the code detect and correct, respectively?

(b) Compute the error probability, i.e., the probability that the decoder cannot make the correct decision.

(c) Compare the error probability with that resulting from soft-decisions.

Chapter 4

Medium Access Control

4.1 Introduction

In the previous chapter, we studied that the physical layer is responsible for the transmission of the message bit stream over the wireless channel. Several issues, e.g., the way information is modulated and transmitted over the channel and the probability of successful message reception were covered. The question we would like to answer in this chapter is when a node gets the right to transmit messages over a wireless communication medium that is shared by many transmitters? Moreover, what is the mechanism to get such a right? This mechanism is regulated by the Medium Access Control (MAC). In this chapter, the main aspects of the MAC layer in WSNs are studied. Within the OSI stack model, the MAC layer is considered as a part of the Data Link layer (DLL).

The principal objective of the MAC layer is to ensure reliable data transmission across the link that the physical layer has already determined. Furthermore, MAC layer determines the way access is controlled in the communication channel, a fundamental function in case of broadcast WSNs where the physical medium is shared by a large number of sensors. Specifically, in any broadcast network the important issue is how to determine which node uses the wireless channel at which time and over which frequency. Therefore, message transmission regulation is needed to achieve an efficient channel allocation amongst the nodes.

MAC layer and its associated protocols that set the rules for the communication between the sending and the receiving node, refer mainly to mechanisms that control the timing of frequency intervals for sending a message (packet) through the channel and listening for it.

The chapter is organized as follows:
Important classes of useful MAC protocols are schedule-based (Section 4.3.1) and contention-based (Section 4.3.2) protocols. In Section 4.4, the IEEE 802.15.4 MAC protocol is discussed. This protocol combines contention and

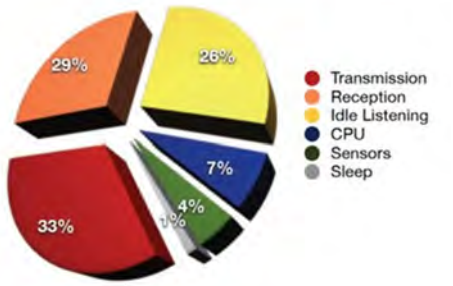


Figure 4.1 Typical power consumption of a node of a WSN.

schedule-based elements and can be expected to achieve significant commercial impact.

4.2 Problems and Performance Requirements for MAC Protocols

4.2.1 Energy Efficiency

The issue of energy efficiency is the prime consideration in WSN MAC protocols. MAC is one of the major components for energy expenditure in WSNs. The power consumption of a node is distributed over its different operations as shown in Figure 4.1. From the figure, we see that about 30% of the power consumption is for message reception, 30% is for message transmission, and 30% for idle listening for incoming messages. The design of MAC protocols for WSNs has to take into account these typical figures. Most of the sensor nodes are deployed with batteries and have limited lifetime, hence prolonging lifetime is of crucial importance. MAC protocols aim to deal with all the operations related to energy consumption and try to minimize their effect. As shown in Figure 4.1, receiving messages is about as expensive as transmitting, and idle listening for messages consumes also a significant amount of a sensor's energy.

Typical sources of energy wastage in WSNs include message collisions, message overhearing, idle listening and protocol overhead:

- *Frame collisions*: A data message collision occurs when a node sends a data message that collides or overlaps in time with another message. The collision is indicated by a failure from the receiver to return an acknowledgement (ACK) message for the data message. When a message collision occurs the node has to retransmit the lost message to increase the probability that it is successfully received. This activity requires a new session of link establishment between nodes as well as energy for

retransmitting the message. A protocol may reduce message collisions by employing contention-free scheduling protocols or contention- based backoff algorithms to minimize the probability of collisions.

- *Message overhearing*: In this case, effort is consumed in receiving a message destined for another node. Receiving and discarding messages intended for other nodes is commonly employed in non-energy constrained networks to increase throughput and decrease latency. Energy-efficient methods to deal with this problem are early rejection and network allocation vector (NAV) sleep. Specifically, early rejection allows a sensor node to turn off its radio once it has read a different destination field for an incoming message. Furthermore, NAV sleep allows nodes to schedule a sleep period during the overheard request-to send/clear-to send (RTS-CTS) handshake sequence by noting the message duration field and scheduling a NAV table interrupt.

- *Idle listening*: Idle listening occurs when a device listens to a wireless channel that is free from transmissions (idle medium). Contention-based WSN MAC protocols attempt to synchronize network traffic so that transmissions begin only in predetermined time slots. Once all network transmissions are complete for a particular cycle or time duration of a message, the protocols allow nodes to return to sleep until the next transmission period. Contention-free WSN MAC protocols reduce idle listening by scheduling transmission slots and allowing nodes not actively exchanging messages to sleep.

- *Protocol overhead*: The overhead in MAC protocols can result form per-message overhead (MAC headers and trailers), collisions, or from exchange of extra control messages. When designing a MAC protocol one should keep the number of control messages to a minimum. Otherwise this overhead will result in unnecessary energy consumption while transmitting and receiving.

As we saw in the previous chapter, the received power decreases with the distance between transmitting and receiving node. This path loss combined with that any transceiver needs a minimum signal strength to demodulate signals successfully leads to a maximum range that a sensor node can reach with a given transmit power. If two nodes are out of reach, they cannot hear each other. This gives rise to the well-known hidden terminal/exposed terminal problems that are described in the following sections. In general, there are three types of ranges:

- *Transmission Range*: The range within which a message can be successfully received. This value is mainly determined by the transmission power, the receiver sensitivity (SNR) threshold requirement and the radio propagation environment.

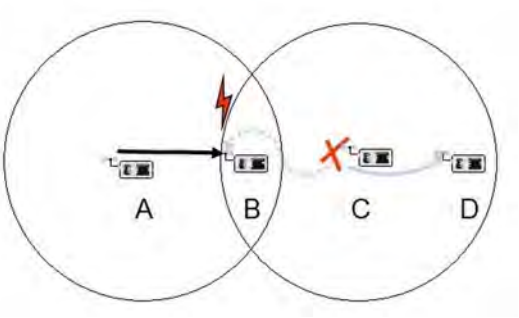


Figure 4.2 The hidden terminal problem. The transmitter A does not hear the transmissions of C, which results in a collision in B.

- *Carrier Sense Range*: For a sending node, Carrier Sense Range is defined as the range within which all other nodes will detect the channel busy. It is determined by the power sensing threshold, the antenna sensitivity, and the wireless channel propagation properties.
- *Interference Range*: For a receiving node, Interference Range indicates the range within which an unrelated transmitter can corrupt the message at the receiver.

4.2.2 The Hidden Terminal Problem

The hidden terminal (here terminal is synonymous of node) problem occurs when a node senses the medium (i.e., the wireless channel) before starting to transmit a message. In wireless networking, the hidden terminal problem occurs when a node is visible from a receiver node, but not from other nodes communicating with said receiver node. The scenario, as shown in Figure 4.2, is as follows: Suppose that node A wants to send a message to node B. Node A does not hear transmitter C sending messages that can be received by node B and node D. The receiving node B is simultaneously included in the transmission range of node A and the interference range of node C, as shown in Figure 4.2. Hence, a collision is possible to occur at B when A and C transmit messages at the same time as B. This collision cannot be directly detected since a carrier-sensing operation by C shows an idle wireless channel because A and C cannot hear each other.

The hidden terminal problem is particularly relevant for Carrier Sense Multiple Access (CSMA) protocols, which we will see later below.

4.2.3 The Exposed Terminal Problem

The exposed terminal problem is shown in Figure 4.3. According to this scenario, node B wants to send a message to node A. In addition, node C

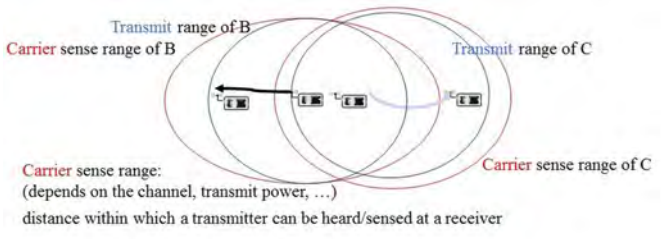


Figure 4.3 The exposed terminal problem: B hears a transmission of C and does not transmit, even though the transmission of B does not cause collisions at A or D.

wants to send a message to node D. Although this would be theoretically possible since both A and D would receive their messages without distortions, the carrier-sense operation performed by node C, suppresses C's transmission and bandwidth is wasted.

4.2.4 Characteristics of MAC Protocols

The essential task of any MAC protocol is to regulate the access of the nodes to the wireless channel so that that certain application-dependent performance requirements are satisfied. Some of the traditional performance criteria are transmission delay, throughput and fairness, whereas in WSNs, the additional performance creterion of energy conservation is important. The most common characteristics of the MAC protocols can be summarized as follows:

- *Transmission delay*: Transmission delay is defined as the amount of time that a single message spends in the MAC protocol. It can by categorized in deterministic delay (related to a predictable number of state transitions) and probabilistic delay that allows only an approximation of the delay, giving the possibility to calculate relative worst or best case bound. The delay issue requires MAC protocols to be simple and have as fewer mechanisms as possible. Designing principles have to compromise on simplicity and low delay with the error control, retransmissions and collision avoidance.

- *Throughput*: Throughput is defined by the rate at which messages are served. The throughput can be measured in messages or symbols per second but most commonly is measured in bits per second. The goal is to maximize it.

- *Fairness*: A MAC protocol is considered fair if it allocates a channel among the competing nodes according to some fairness criteria. How-

ever, fairness is a complex term in WSN. WSN can be viewed as a distributed system and envisaging it so the ratio of channel allocation between nodes may or may not be fair.

- *Scalability*: Scalability describes the ability of the communication system to meet performance characteristics despite of the size of the network and number of competing nodes. Although this metric belongs more to the networks architecture the designer of a MAC protocol has to consider how to handle competition for channel access, retransmission and what happens if the traffic load increases because of the increase of the number of nodes.

- *Robustness*: Robustness is referred to as a composition of reliability, availability, and dependability. It describes protocol sensibility for traffic load over a sustained period of time.

- *Stability*: Stability describes how good the protocol handles fluctuation of traffic load over a sustainable period of time.

4.3 Definition and Classification of MAC Protocols

Depending on the way MAC protocols regulate access on the shared medium, they can be classified into two broad classes: schedule-based (or contention-free) and contention-based protocols. Protocols belonging to the first class, allow only one node at a time to access the wireless channel. A schedule regulates which node may use which time or frequency slot at which time. The schedule can be fixed or computed on demand resulting to a further classification into fixed-assignment and on-demand protocols, respectively. In this case, collisions, overhearing and idle listening are avoided, but time synchronization among nodes is needed.

On the other hand, contention-based protocols allow nodes to access the wireless channel simultaneously. The main principle of these protocols is random access. For this reason, mechanisms are implemented to handle or reduce the occurring message collisions. MAC protocols that do not fit into this classification having characteristics of both contention-free and contention-based techniques are hybrid approaches often aiming to inherit the advantages of these main categories, while minimizing their weaknesses.

4.3.1 Schedule-based MAC Protocols

Typical protocols of this class are TDMA, FDMA and CDMA. The Time Division Multiple Access (TDMA) scheme divides the time axis into slots. These time slots are assigned to each node exclusively and therefore every node transmits periodically only in its own time slot. A In most cases, a central node decides the TDMA schedules. Synchronization is also needed

among the nodes to avoid message collisions in adjacent time slots. This scheme is useful in small networks or when the network is divided into smaller clusters, where, in each of them, MAC can be controlled at a local cluster head.

In TDMA MAC protocols, we assume that N time slots are assigned to N nodes in the network. The nodes are synchronized with a predetermined synchronization message transmission frequency f_s. Each slot is long enough to accommodate guard time to account for synchronization errors and one message transmission. The length of guard time is a function of f_s.

The average delay to transmit a message is given as follows:

$$D = \frac{N}{2} t_s + t_p = \frac{N}{2} \left(\frac{d}{f_s} + 2t_p \right) , \tag{4.1}$$

where t_s is the duration of TDMA slot, d is the clock drift per unit time and t_p is the message transmission time, which corresponds to L times the slot time r_s where L denotes the message transmission duration measured in slots[1] and r_s the duration of the slot.

The reliability of TDMA schemes is approximately 1 assuming appropriate transmit radio powers are used and ignoring the interference from other networks.

The average energy consumption is given as follows:

$$E = P_r L r_s f_s + P_t \frac{L}{L+X} + P_s \left(1 - \frac{L}{L+X} - L r_s f_s \right) , \tag{4.2}$$

based on the assumption that $L/(L+X) + L r_s f_s \leq 1$. X denotes the time duration to wait before the next transmission attempt measured in slots and P_r, P_t and P_s are the average energy consumption in receive, transmit and sleep states respectively.

In Frequency Division Multiple Access (FDMA), the available frequency band for transmissions is divided into subchannels, each of which is assigned to particular node. Since each node has its own private frequency band, centered around a carrier frequency, there is no interference between different nodes. In this case, frequency synchronization as well as narrowband filters are required.

Code Division Multiple Access (CDMA) assigns a different code to each node. Each node then uses its unique code to encode the data bits it sends. If the codes are chosen carefully, different nodes can transmit simultaneously and yet have their respective receivers correctly receive a sender's encoded message bits (assuming the receiver knows the sender's CDMA code and the codes are "orthogonal") in spite of interfering transmissions by other nodes.

The above mentioned schedule-based methods are efficient and provide better performance when the number of nodes in the network is small and

[1] We assume that this duration is an integer number of slots.

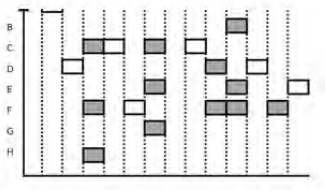


Figure 4.4 The slotted ALOHA protocol.

constant. On the contrary, when the number of nodes is large and continuously varying or the traffic is bursty, schedule based methods present some problems. In specific, in TDMA, delay occurs whenever a user does not use the allocated slot. In FDMA, if the carrier frequency is divided into N slots and fewer than N nodes are currently interested in communicating, a large piece of valuable spectrum is wasted. Furthermore, if more than N nodes want to communicate, some of them will be denied permission due to lack of bandwidth, even if some of the sensors that have been assigned a frequency band hardly ever transmit or receive anything.

4.3.2 Contention-based MAC Protocols

When the number of nodes is large and variable or the traffic is fairly bursty, schedule-based schemes are poor choices. In this case, certain contention-based MAC protocols have been proposed as alternative. Typical protocols of this class are the ALOHA and CSMA protocols, which we study below.

ALOHA Protocols

Slotted ALOHA is considered as one of the simplest contention-based MAC protocols. It works on top of TDMA as time is divided into slots of size equal to the time interval a message requires to be transmitted. In this scheme, nodes are synchronized and start to transmit messages only at the beginnings of the slots. In case that two or more messages collide during a slot, then all the nodes detect the collision event before the slot ends. In particular, let p be a probability that a node can transmit a message. The steps of slotted ALOHA (Figure 4.4) are the following:

- When a node has a new message to send, it waits until the beginning of the next slot and transmits the entire node in the slot.
- If no collision occurs, the node has successfully transmitted its node and thus need not consider retransmitting the message.
- If collision occurs, the node detects the collision before the end of the slot and retransmits its message in each subsequent slot with probability p until the message is transmitted without a collision.

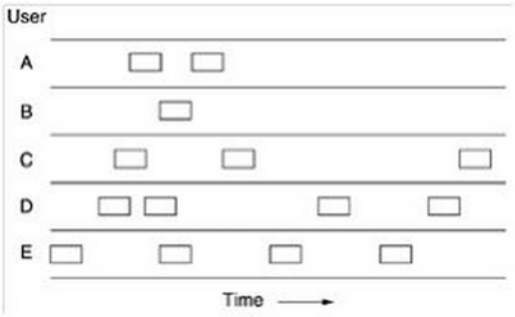


Figure 4.5 The pure ALOHA protocol.

Slotted ALOHA constitutes a simple MAC protocol. It is also highly decentralized, because each node detects collisions and independently decides when to retransmit. Let's assume that each sensor always has a message to send and that it transmits with probability p for a new message as well as for a message that has already suffered a collision. Suppose also that n is the number of nodes attempting to transmit. Then the probability that a given slot is a free slot, is given by the probability that one of the nodes transmits and that the remaining $n-1$ nodes do not transmit. The probability that a given node transmits is p; the probability that the remaining nodes do not transmit is $(1-p)^{n-1}$. Therefore the probability a given node has a success is $p(1-p)^{n-1}$. Because there are n nodes, the probability that any one of the n nodes has a success, or the probability that a slot is taken, is

$$\left(\begin{array}{c} n \\ 1 \end{array} \right) p(1-p)^{n-1} .$$

The slotted ALOHA protocol requires that all nodes synchronize their transmissions to start at the beginning of a slot. The first ALOHA protocol was actually an unslotted, fully decentralized protocol, a so called pure ALHOA. In pure ALOHA, messages are transmitted at completely arbitrary times and the basic idea is simple: nodes transmit whenever they have messages to send. If a transmitted message experiences a collision, which is found out either by listening to the channel at the receiver node or by negative acknowledgement messages (NACK) from the receiver to the transmitter node, the node will then immediately (after completely transmitting its collided message) retransmit the message again with probability p. Otherwise, the node waits for a random amount of time and it then retransmits the message with probability p. The waiting time must be random or the same messages may collide over and over, in lockstep. This is an unsynchronized transmission at any instant, since time is not divided into slots during which messages must fit, as it is shown in Figure 4.5.

CSMA Protocols

In both slotted and pure ALOHA, a node's decision to transmit a message is made independently of the activity of the other nodes sharing the wireless channel. In Carrier Sense Multiple Access (CSMA) protocols, a node first listens (channel assessment) if the channel is free or busy from other transmissions.

Some protocols of this class may also be characterized by the collision detection property (CSMA/CD). In specific, a transmitting node listens to the channel while it is transmitting. If it detects that another node is transmitting an interfering message, it stops transmitting and waits a random amount of time before repeating the listen-and-transmit-when-idle cycle. CSMA/CD protocols are most applicable in case of a wired medium where an absence of collision at the transmitting node means that there will be also no collision at the receiver node during message transmission. However, in a wireless channel, collisions may occur at the receiver and the transmitting node will therefore be unaware of a collision.

Another important class of CSMA protocols are those equipped with the property of collision avoidance. In the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocols, nodes attempt to avoid collisions by not transmitting immediately when the channel is sensed to be idle. In particular, a node, first, listens to the channel for a specific time, which is generally referred to as the intermessage space (IFS) and which is a multiple of a slot size. The IFS duration ensures that nodes transmit only if the channel is idle and helps to prevent collisions. If the channel is sensed as busy during the IFS, it is determined that another node is transmitting a message. Therefore, the competing nodes will defer their transmission until the end of ongoing transmission. However, at the end of the ongoing transmission, the competing nodes would sense the channel as idle and try to send their messages simultaneously. In order to prevent this collision, random backoff periods are assigned to the competing nodes before they transmit a message.

This backoff mechanism of CSMA/CA protocols is crucial for the avoidance of collisions. Once the ongoing transmission is over, the nodes delay another IFS. If the wireless channel remains idle for this period, the nodes pick up a random number of slots within a range of values to wait before transmitting their own message. This range of values is referred to as the contention window and nodes select a random number within this window for their backoff. The backoff is performed through a timer, which reduces the backoff value for each specific duration equal to a slot. After the nodes enter the backoff period, the first node with its clock expiring starts transmission. In other words, the node with the shortest backoff period starts to send while the other nodes sense the new transmission and freeze their backoff clocks, to be restarted after completion of the current transmission in the next con-

tention period. This random backoff mechanism aims to prevent nodes from self-synchronizing at the end of a transmission and colliding with each other. However, it is clear that in the case of dense networks, there will be many nodes that will enter the backoff mechanism. As a result, with some probability, some nodes can select the same backoff period and collide with each other. In this case, the nodes that collide double their contention window and select a new backoff period. This mechanism is referred to as the binary exponential backoff scheme. In the case of a successful communication, the contention window is set to its initial value.

Other schemes equipped with the collision avoidance property rely on a dynamic channel reservation mechanism. Specifically, nodes send small control messages to reserve the wireless channel prior to data transmission. The channel reservation is performed through two small control messages, i.e. request-to-send (RTS) and clear-to-send (CTS) control messages as in Multiple Access with Collision Avoidance (MACA). With an RTS message, a node indicates its desire to transmit a data message to an intended receiver. If the RTS arrives without collision the receiver node responds with a CTS control message, granting the right to send the message. If a sender does not receive a CTS in response to its RTS, it will retry at a later time. However, if the CTS message has been received, the channel reservation has concluded successfully. The competing nodes, hearing either the RTS or CTS message, are informed about the ongoing transmission and they wait before they attempt to reserve the channel. Using this handshake, MACA addresses the hidden terminal problem and reduces the number of collisions by reserving the medium for data transmissions.

In MACA, this waiting time can be based upon the size of the data transmission, which can be indicated as part of the RTS and CTS messages. This introduces the Network Allocation Vector (NAV) process according to which when a node sends an RTS message, the duration of the data message that will be sent is also indicated in the RTS message. Moreover, the receiver node replies to this RTS with a CTS message, copying the duration of the data message. As a result, nodes that hear either the RTS or CTS message can determine the duration of the channel reservation handshake. These nodes can refrain from continuously sensing the channel during the transmission. NAV consists of the duration of the transmission that is being performed and is reduced in every slot just as the backoff timer. Physical channel sensing is only performed when NAV expires. As indicated previously, one of the major problems of the CSMA/CA scheme is the requirement for continuous channel sensing. NAV helps, to some extent, to reduce the energy consumed for channel sensing by enabling the nodes to sleep during an ongoing transmission until NAV expires. The NAV concept is one of the most exploited ideas in MAC protocols for WSNs.

In MACAW (MACA for Wireless LANs), the receiver responds with an acknowledgement (ACK) control message once the message has been received

correctly, allowing other nodes to learn that the channel is available again and to increase the reliability of transmissions. The consequence is that nodes overhearing an RTS message must remain silent to ensure that the sender of the RTS is able to receive the ACK. Nodes that overheard RTS, but did not acknowledge CTS, do not know whether they did not hear the CTS signal because they are out of reach of the destination or because the CTS message was never sent. In either case, they will also not hear the ACK from the destination, that is, they must stay silent until the expected completion time of the transmission, based on the information carried in the RTS message. However, if no CTS was sent, they remain silent and delay their own transmission, even though no interfering transmissions are occurring.

Therefore, the MACAW protocol introduces another control message, called the data sending (DS) message. The DS message is sent by the node issuing the RTS message after receiving the corresponding CTS message to confirm that a transmission will actually take place. A node overhearing an RTS message, but not the corresponding DS message, may assume that the medium reservation has failed and can attempt to reserve the wireless channel for its own communication.

4.4 The IEEE 802.15.4 Standard for WSNs

In this section, the architecture and the basic principles of IEEE 802.15.4 standard are summarized with specific reference to the MAC. IEEE 802.15.4 is the de-facto reference standard for low data rate and low power WSNs. The IEEE 802.15.4 standard specifies two layers: the physical and MAC layer.

4.4.1 Overview

The main characteristic of IEEE 802.15.4 is the low data rate for ad hoc self-organizing networks of inexpensive fixed, portable and moving nodes. In the standard, the nodes are called “devices”. The standard provides high network flexibility and very low power consumption. The standard introduces a MAC with a superframe structure with two consecutive periods, i.e., contention access period and contention-free period.

The network is assumed to be clustered and each cluster head, i.e., Personal Area Coordinator (PAN) coordinator, broadcasts the frame structure and allocates slots to prioritized traffic in the contention-free period.

In the contention period, nodes contend using either CSMA/CA or slotted CSMA/CA to access the wireless channel. The winners can allocate the channel for their transmissions for a specific amount of time. This provides a flexible access method for nodes with infrequent traffic.

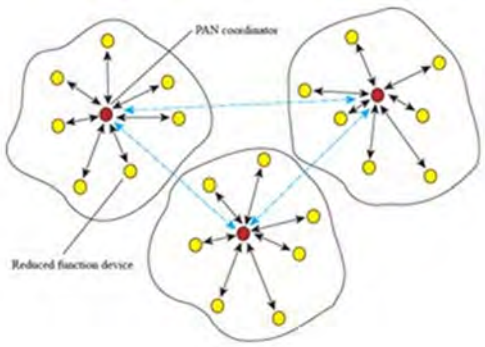


Figure 4.6 An IEEE 802.15.4 WSN.

During the contention-free period, nodes with higher priority traffic are served by the PAN coordinator. Based on the traffic requirements, each node is assigned slots during the contention-free period. These slots are allocated to only one pair and channel contention is prevented to provide priority. As a result, the IEEE 802.15.4 protocol provides a hybrid operation through a CSMA-based and a TDMA-based operation. Section 4.4.2 summarizes the main features of an IEEE 802.15.4 network. The IEEE 802.15.4 physical and MAC layer are described in sections 4.4.3 and 4.4.4, respectively.

4.4.2 An IEEE 802.15.4 Network

An IEEE 802.15.4 network is composed of two different kinds of network devices; full-function devices (FDD) and reduced-function devices (RFD). The network includes at least one FFD which can operate in three modes; as a personal area network (PAN) coordinator, as a simple coordinator or as a simple device. An FFD can talk to RFDs or FFDs whereas an RFD can only talk to an FFD. RFDs are destined for simple applications and they are involved in transmissions of small amount of data. Figure 4.6 shows a typical IEEE 802.15.4 network. An IEEE 802.15.4 network can operate in three different topologies; Star topology, peer-to-peer topology and cluster tree topology.

In the star topology, the communication is established between the devices and a central controller that is called PAN coordinator. A PAN coordinator, besides the application that needs to perform, is responsible for critical network tasks, such as the start, the termination and the routing of the communication. In this type of topology, the PAN coordinator should be constantly connected to a power supply whereas the rest devices can have batteries as their source of energy. A local network organized in a star topology is characterized by a unique PAN identifier number that allows it to operate independently from all the other local networks within its

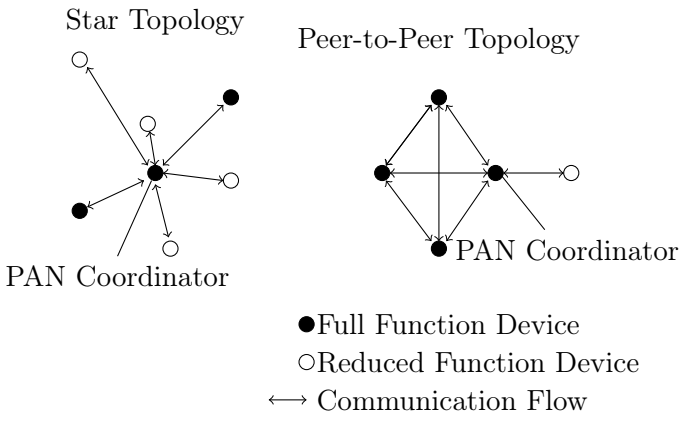


Figure 4.7 Star and peer-to-peer topology of IEEE 802.15.4.

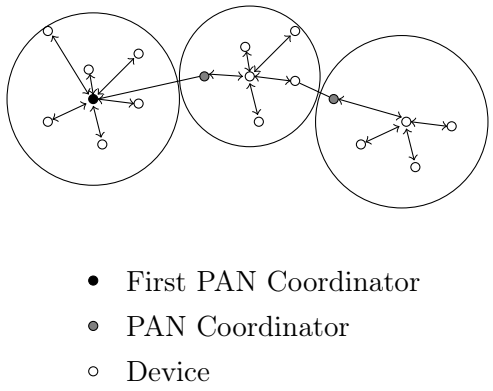


Figure 4.8 The cluster-tree topology of IEEE 802.15.4.

transmission range.

The concept of PAN coordinator exists also in the case of a peer-to-peer topology. The main difference with respect to the star topology is that now all the devices are able to communicate each other as far as a device is located inside the range of another. This sort of topology also allows network implementations with higher complexity level and, therefore, it is very popular in WSN applications as it enables self organization and self configuration (ad hoc networks). These two topologies are shown in Figure 4.7.

Cluster-tree topology, as shown in Figure 4.8, is a special case of a peer-to-peer topology where every FFD is able to operate as a coordinator and provide synchronization both to other devices and to other coordinators. In this topology, an RFD is connected only at the end of a cluster branch as it is able to communicate with only one FFD at a time. Moreover, the

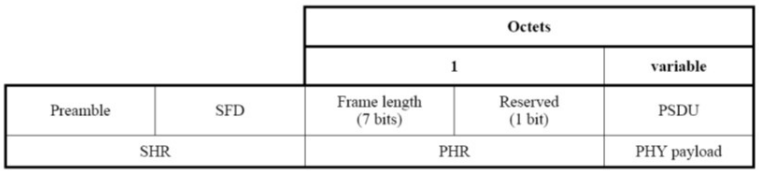


Figure 4.9 Physical layer data unit of IEEE 802.15.4.

coordinator of the first cluster operates as a global PAN coordinator and consumes most of the computational network resources with respect to any other device. The main advantage of this particular topology is the wide coverage of an area; however, the propagation speed of the messages remains low.

4.4.3 Physical Layer

The IEEE 802.15.4 physical layer provides two main services; the data service and the management service. Data service allows transmitting and receiving packets (physical layer data units) across the wireless channel. Figure 4.9 shows the typical structure of a physical layer data unit. Each packet consists of the following basic components:

- SHR, which allows a receiving device to synchronize and lock into the bit stream
- PHR, which contains frame length information
- a variable length payload, which carries the MAC sublayer frame

Start of Frame Delimiter (SFD) indicates the end of the SHR and the start of the message data. The preamble field that is used for synchronization together with the SFD field form the SHR (SyncH) header. PHR header indicates the length of the PSDU (PHY Service Data Unit) payload which has a non constant value (<128 byte).

The main features of the physical layer include the activation and deactivation of the radio transceiver, the energy detection (ED, from RSS), the link quality indication (LQI), the clear channel assessment (CCA) and the dynamic channel selection by scanning a list of channels in search of a beacon. ED process includes an evaluation of the received signal's power and the result is stored in order to be used by higher layers. LQI determines a procedure according to which, when a packet is received in the physical layer, an evaluation of its quality is performed based on the value of ED. Moreover, during CCA, the channel is checked in order to identify if it is busy or idle. This becomes possible by checking whether the ED value has

	Frequency band	Coverage	Channels	Data rate
Physical layer	2.4 - 2.4835 GHz	Global	16	250 Kbps
	902.0 - 928.0 MHz	America	10	40 Kbps
	868 - 868.6 MHz	Europe	1	20 Kbps

Table 4.1 Characteristics of IEEE 802.15.4 physical layer.

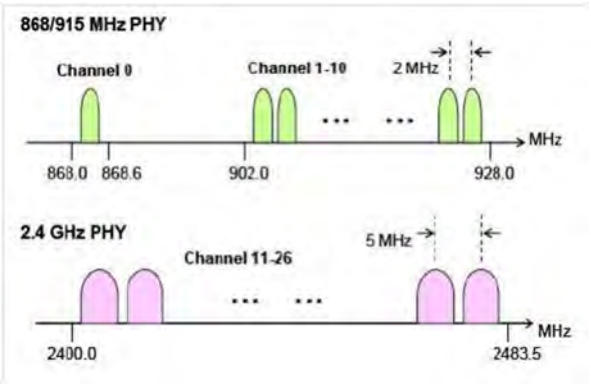


Figure 4.10 Frequency channels of the IEEE 802.15.4 standard.

overcome a particular limit value, by detecting the carrier or by detecting a signal with the expected modulation scheme in the channel. The result is also stored and can be used by higher layers. Table 4.1 provides the main characteristics of IEEE 802.15.4 physical layer.

The standard defines three different frequency bands in which it can operate. The three frequency bands result in a total of 27 frequency channels (Figure 4.10), numbered from 0 to 26. 16 channels are available in the frequency band of 2450 MHz, 10 channels are available in the band of 915 MHz and 1 channel in the band of 868 MHz. The central frequency of these channels is determined as follows, where k represents the channel number:

$$F_c = 868.3 \text{ MHz with } k = 0 \tag{4.3}$$

$$F_c = 906 + 2(k-1) \text{ MHz with } k = 1, 2 \ldots, 10 \tag{4.4}$$

$$F_c = 2405 + 5(k-11) \text{ MHz with } k = 11, 12, \ldots, 26 \tag{4.5}$$

Any particular device can operate in different frequency bands with data and spreading parameters as indicated in Table 4.2.

Table 4.2 Frequency bands and propagation parameters for IEEE 802.15.4 physical layer.

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868-868.6	300	BPSK	20	20	Binary
	902-928	600	BPSK	40	40	Binary
868/915 (optional)	868-868.6	400	ASK	250	12.5	20-bit PSSS
	902-928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868-868.6	400	O-QPSK	100	25	16ary Orthogonal
	902-928	1000	O-QPSK	250	62.5	16ary Orthogonal
2450	2400-2483.5	2000	O-QPSK	250	62.5	16ary Orthogonal

4.4.4 MAC Layer

The IEEE 802.15.4 MAC layer provides two services; data service and management service. The data service allows transmitting and receiving packets (MAC layer data units) through the interaction with the data service of the IEEE 802.15.4 physical layer. The main features of the MAC layer include the beacon management, the channel access, GTS (Guaranteed Time Slots) management, frame validation, acknowledged frame delivery and the association and disassociation with the PAN. There are two different channel access mechanisms. In the contention-based access mechanism (beacon-enabled network) a Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) algorithm is implemented by the devices at the MAC layer. On the other hand, the access without contention (non beacon-enabled network) is exclusively controlled by the PAN coordinator by appropriate allocation of the GTSs. This distinction is described in the following figure.

Superframes

The IEEE 802.15.4 standard allows the optional use of the superframe structure. The format of a superframe is determined by the PAN coordinator. It is normally bounded by two network beacon frames and divided into 16 equally sized slots. The beacon frame is sent in the first slot of each superframe. If a coordinator does not want to use the superframe structure, it may turn off the beacon transmissions. The beacon frames are used in order to synchronize the attached nodes, identify the PAN and describe the structure of the superframes. Beacons are sent during the first slot of each superframe and they are turned off if a coordinator does not use the superframe structure. As shown in Figure 4.12, a superframe is divided into two different time portions. During the active portion, communication is performed. In particular, the active period is further divided into two periods; the contention access period (CAP) where any device wishing to communi-

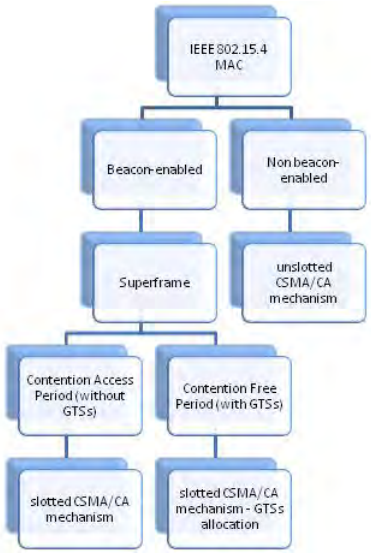


Figure 4.11 MAC options in IEEE 802.15.4.

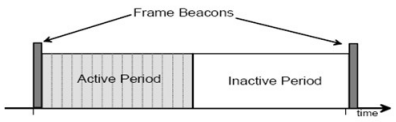


Figure 4.12 Superframe structure of IEEE 802.15.4.

cate competes with other devices using a slotted [2] CSMA/CA mechanism and the contention free period (CFP) which contains guaranteed time slots (GTSs). The GTSs always appear at the end of the active superframe starting at a slot boundary immediately following the CAP. The PAN coordinator may allocate up to 7 GTSs where each of them can occupy more than one superframe slot periods as shown in Figure 4.13. A GTS allows a device to operate within a portion of the superframe that is dedicated exclusively to it. A device attempts to allocate and use a GTS only if it is tracking the beacons. As far as the GTS allocation is concerned, it is undertaken by the PAN coordinator only. A GTS is used only for communications between the PAN coordinator and a device and its direction is specified as either transmit or receive. On the contrary, in the inactive portion, a node does not interact with its PAN and may enter a low-power mode.

The duration of different portions of the superframe are described by the

[2] If a superframe structure is used in the PAN, then slotted CSMA/CA is used in the CAP period. On the contrary, if beacons are not used in the PAN, or a beacon cannot be located in a beacon-enabled network, unslotted CSMA/CA is used.

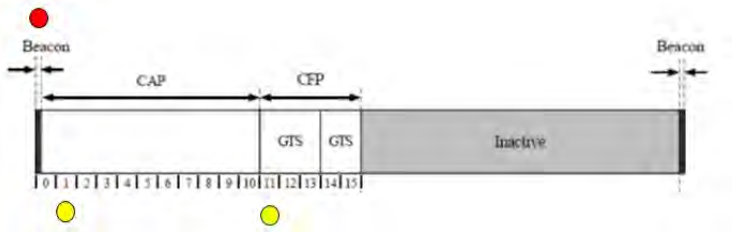


Figure 4.13 GTSs in a IEEE 802.15.4 superframe.

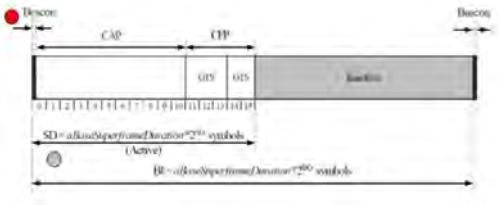


Figure 4.14 Duration of different portions of the IEEE 802.15.4 superframe.

values of *macBeaconOrder* and *macSuperFrameOrder*. The value of *macBeaconOrder* represents the interval at which the PAN coordinator shall transmit its beacon frames. The beacon interval, BI, is related to the *macBeaconOrder*, BO, as follows:

$$BI = aBaseSuperFrameDuration 2^{BO}, 0 \leq BO \leq 14. \quad (4.6)$$

The superframe is ignored if BO = 15. The value of *macSuperFrameOrder* describes the length of the active portion of the superframe. The superframe duration, SD, is related to *macSuperFrameOrder*, SO, as follows:

$$SD = aBaseSuperFrameDuration 2^{SO}, 0 \leq SO \leq 14. \quad (4.7)$$

If SO = 15, the superframe should not remain active after the beacon. The active portion of each superframe is divided into a *aNumSuperFrameSlots* equally spaced slots of duration $2^{SO} aBaseSlotDuration$ and is composed of three parts: a beacon, a CAP and CFP. The beacon is transmitted at the start of slot 0 without the use of CSMA. The CAP starts immediately after the beacon. The CAP shall be at least *aMinCAPLength* symbols unless additional space is needed to temporarily accommodate the increase in the beacon frame length to perform GTS maintenance. All frames except acknowledgement frames or any data frame that immediately follows the acknowledgement of a data request command that are transmitted in the CAP shall use slotted CSMA/CA to access the channel. A transmission in the

CAP shall be complete one IFS[3] period before the end of the CAP. If this is not possible, it defers its transmission until the CAP of the following superframe. An example superframe structure is shown in Figure 4.14. The CFP, if present, shall start on a slot boundary immediately following the CAP and extends to the end of the active portion of the superframe. The length of the CFP is determined by the total length of all of the combined GTSs. No transmissions within the CFP shall use a CSMA/CA mechanism. A device transmitting in the CFP shall ensure that its transmissions are complete one IFS period before the end of its GTS. The PANs that do not wish to use the superframe in a nonbeacon-enabled shall set both *macBeaconOrder* and *macSuperFrameOrder* to 15. In this kind of network, a coordinator shall not transmit any beacons, all transmissions except the acknowledgement frame shall use unslotted CSMA/CA to access channel, GTSs shall not be permitted.

CSMA/CA Mechanism

An IEEE 802.15.4 network uses two different approaches regarding the channel access according to the network settings. More specifically, the networks that do not use beacon frames or the beacon frames cannot be located in a beacon-enabled modality, provide an unslotted CSMA/CA mechanism and each time a device wants to transmit data frames or MAC layer commands, it waits for a random time period. By the expiration of this period, if the channel is found inactive, the device sends its data; otherwise, if the channel is busy, the device waits for a random time period until it checks again the availability of the channel. The acknowledgement frames are sent without the use of CSMA/CA mechanism.

On the other hand, the networks that use beacon frames within a superframe structure provide a slotted CSMA/CA mechanism on which the back-off slots are located at the start of the beacon frames. Every time a device wants to transmit data frames during the CAP, after specifying the bounds of the next back-off slot, it waits for a random number of back-off slots. By the end of this waiting time, if the channel is busy, the device will wait for a random number of back-off slots until it checks again the availability of the channel. In case that the channel is free, it will start transmitting.

In each transmission attempt, the first step of the CSMA/CA algorithm refers to the initialization of some parameters. Each device is characterized by three particular variables: NB, CW and BE. NB is defined as the number of times the CSMA/CA algorithm was required to back-off while attempting the current transmission. It is initialized to zero before every

[3]IFS is defined as the amount of time necessary to process the received packet by the physical layer.

new transmission attempt. CW is the contention window length (used for slotted CSMA/CA) which represents the number of back-off periods that need to be clear of activity before the transmission can start. It is initialized to 2 before each transmission attempt and it is reset to 2 each time the channel is assessed to be busy. CW is only used in the case of slotted CSMA/CA. BE is the back-off exponent that describes how many back-off periods a device shall wait before attempting to access the channel. The parameters that affect the random backoff are $\mathrm{BE}_{\mathrm{min}}$, $\mathrm{BE}_{\mathrm{max}}$ and $\mathrm{NB}_{\mathrm{max}}$, which correspond to the minimum and maximum of BE and the maximum of NB respectively.

In the second step of the algorithm for the slotted CSMA/CA mechanism, the MAC layer delays for a random number of back-off periods in the interval $[0, 2^{\mathrm{BE}} - 1]$. Moreover, during the third step, MAC layer requires from the physical layer to perform a CCA and it then proceeds only if the transmission of the acknowledgement frames has been completed before the end of CAP. If the MAC layer cannot proceed, it waits until the start of the next superframe's CAP and it repeats the evaluation of the channel. In the fourth step, in case that the channel is evaluated as busy, MAC layer increases the NB and BE variables by one, ensuring that BE is less than its maximum value, $\mathrm{BE}_{\mathrm{max}}$. In slotted CSMA-CA, CW can also be reset to 2. Moreover, if NB value is less than the maximum number of CSMA/CA back-offs, then the MAC layer returns to the second step of the algorithm; otherwise, the channel access attempt fails and CSMA/CA terminates. In case that the channel is evaluated as idle, the MAC layer, in the CSMA/CA slotted mechanism, ensures that CW expires before the transmission begins. For this reason, MAC reduces CW value by one and if CW becomes equal to zero, the transmission begins on the boundary of the next back-off period; otherwise, the algorithm returns to the third step. MAC layer starts the transmission immediately if the channel is evaluated as idle.
The following flow diagram presents the steps of the CSMA/CA algorithm in slotted and unslotted modality.

The unslotted CSMA/CA mechanism can be analysed by the use of the Markov Model as shown in Figure 4.15.

The goal of the analysis is to derive expressions for the probability that a packet is successfully received, the delay in the packet delivery, and the average energy consumption. The analysis requires finding a set of equations that define the optimal network operating point uniquely.

The analysis is developed in two steps. We first study the behavior of a single device by using a Markov model (Fig.4.15). From such a model, we obtain the stationary probability ϕ that the device attempts its carrier channel assessment (CCA). Second, we couple the per user Markov chains to obtain an additional set of equations that give the CCA assessments of other users. The solution of such a set of equations provides us with the per user ϕ and probability of free channel assessment.

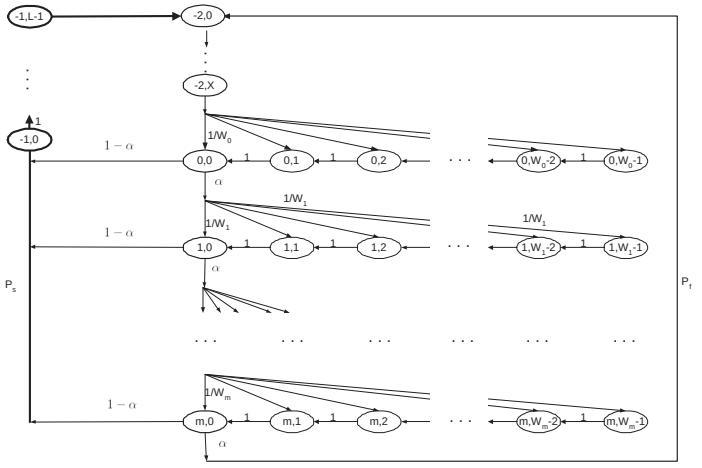


Figure 4.15 Markov Model for IEEE 802.15.4 unslotted random access.

We first develop the Markov model to determine ϕ, see Fig. 4.15. Let $c(t)$ be the stochastic process representing the counter for random delay and packet transmission duration. The integer time t corresponds to the beginning of the slot times. Let α be the probability of assessing channel busy during CCA. Next, when entering the transmission state, L slots should be counted, where L denotes the packet transmission duration measured in slots[4]. Let X denote the time duration to wait before the next transmission attempt measured in slots. Let $s(t)$ be the stochastic process representing the delay line stages representing the number of times the channel is sensed busy before packet transmission ($s(t) \in \{0, \cdots, \mathrm{NB}\}$), or the transmission stage ($s(t) = -1$) at time t. The states ($s(t) = -2$) in Fig. 4.15 model unsaturated periodic traffic. We assume that the probability to start sensing is constant and independent of all other devices and of the number of retransmissions suffered. With this assumption, $\{s(t), c(t)\}$ is the two-dimensional Markov chain of Fig. 4.15 with the following transition probabilities:

$$\mathrm{P}\{i, k|i, k+1\} = 1, \qquad k \geq 0 \tag{4.8}$$

$$\mathrm{P}\{0, k|i, 0\} = \frac{1-\alpha}{W_0}, \qquad i < \mathrm{NB} \tag{4.9}$$

$$\mathrm{P}\{i, k|i-1, 0\} = \frac{\alpha}{W_i}, \qquad i \leq \mathrm{NB}, k \leq W_i - 1 \tag{4.10}$$

$$\mathrm{P}\{0, k|NB, 0\} = \frac{1}{W_0}. \tag{4.11}$$

In these equations, the delay window W_i is initially $W_0 = 2^{BE_{\min}}$ and doubled any stage until $W_i = W_{\max} = 2^{BE_{\max}}, (BE_{\max} - BE_{\min}) \leq i \leq \mathrm{NB}$. Equation 4.8 is the condition to decrement the delay line counter per slot.

[4]We assume that this duration is an integer number of slots.

Equation 4.9 states that it is only possible to enter the first delay line from a stage that is not the last one ($i < \mathrm{NB}$) after sensing the channel idle and hence transmitting a packet. Equation 4.10 gives the probability that there is a failure on channel assessment and the station selects a state in the next delay level. Equation 4.11 gives the probability of starting a new transmission attempt when leaving the last delay line, following a successful or failed packet transmission attempt.

Denote the Markov chain steady-state probabilities by

$$b_{i,k} = \mathrm{P}\{(s(t), c(t)) = (i, k)\}\,,$$

for $i \in \{-1, \mathrm{NB}\}$ and $k \in \{0, \max(L-1, W_i - 1)\}$. Using Equation 4.10, we have

$$b_{i-1,0}\alpha = b_{i,0}\,, \quad 0 < i \leq \mathrm{NB}, \tag{4.12}$$

which leads to

$$b_{i,0} = [\alpha]^i b_{0,0}\,, \quad 0 < i \leq \mathrm{NB}\,. \tag{4.13}$$

From Equations (4.8)–(4.11) we obtain

$$b_{i,k} = \frac{W_i - k}{W_i}\left[(1-\alpha)\sum_{j=0}^{\mathrm{NB}} b_{j,0} + \alpha b_{\mathrm{NB},0}\right] \quad \text{for } i = 0, \tag{4.14}$$

$$b_{i,k} = \frac{W_i - k}{W_i} b_{i,0}, \quad \text{for } i > 0. \tag{4.15}$$

Since the probabilities must sum to 1, it follows that

$$\begin{aligned} 1 &= \sum_{i=0}^{\mathrm{NB}} \sum_{k=0}^{W_i - 1} b_{i,k} + \sum_{i=0}^{L-1} b_{-1,i} + \sum_{i=0}^{X-1} b_{-2,i} \\ &= \sum_{i=0}^{\mathrm{NB}} b_{i,0}\left[\frac{W_i + 1}{2} + (1-\alpha)L + (1-\alpha)X\right] + b_{\mathrm{NB},0}\alpha X. \end{aligned} \tag{4.16}$$

By substituting the expression for W_i, we obtain

$$\begin{aligned} 1 &= \frac{b_{0,0}}{2}\left\{[1 + 2(1-\alpha)(L+X)]\frac{1-\alpha^{\mathrm{NB}+1}}{1-\alpha}\right. \\ &\quad + 2X\alpha^{\mathrm{NB}+1} + 2^{\mathrm{DIFFBE}} W_0 \frac{\alpha^{\mathrm{DIFFBE}+1} - \alpha^{\mathrm{NB}+1}}{1-\alpha} \\ &\quad \left. + W_0 \frac{1 - (2\alpha)^{\mathrm{DIFFBE}+1}}{1-2\alpha}\right\}, \end{aligned} \tag{4.17}$$

where DIFFBE = $\mathrm{BE}_{\max} - \mathrm{BE}_{\min}$. The transmission failure probability P_f is

$$\mathrm{P}_f = b_{\mathrm{NB},0}\alpha\,, \tag{4.18}$$

and the probability that a node starts to transmit is

$$\tau = \mathrm{P}_s = \phi(1-\alpha)\,, \tag{4.19}$$

in which

$$\phi = \phi_1 = \sum_{i=0}^{\mathrm{NB}} b_{i,0} = b_{0,0}\frac{1-\alpha^{\mathrm{NB}+1}}{1-\alpha}\,. \tag{4.20}$$

We have now derived one expression for ϕ from the per user Markov models. By determining the interactions between users on the medium, we will now derive expressions for α. Assume that there are N nodes in the network. Denote by $M(s) = -1$ the event that there is at least one transmission in the medium by another node and assume that, without loss of generality, the sensing node is i_N, which is denoted as $S^{i_N}(c) = -1$ if $S^i(s) = -1$ is the event that node i is transmitting. Then, the probability that a node sensing the channel finds it occupied is $\alpha = \Pr(M(s) = -1 | S^{i_N}(c) = -1)$, which is computed as follows

$$\begin{aligned}
\alpha =& \Pr(M(s) = -1 | S^{i_N}(c) = -1) \\
=& \sum_{n=0}^{N-2} \left(\begin{array}{c} N-1 \\ n+1 \end{array} \right) \Pr\left(\bigcap_{k=1}^{n+1} S^{i_k}(s) = -1 | S^{i_N}(c) = -1 \right) \\
=& \sum_{n=0}^{N-2} \left(\begin{array}{c} N-1 \\ n+1 \end{array} \right) \Pr\left(S^{i_1}(s) = -1 \right) \\
& \times \Pr\left(\bigcap_{k=2}^{n+1} S^{i_k}(s) = -1 | S^{i_1}(s) = -1, S^{i_N}(c) = -1 \right)\,.
\end{aligned} \tag{4.21}$$

The probability that node i_1 is transmitting is

$$\Pr(S^{i_1}(s) = -1) = (L+1)P_s = (L+1)\phi(1-\alpha)\,, \tag{4.22}$$

which requires the node to sense (with probability ϕ) before transmission and the following slot to be empty (with probability $(1-\alpha)$). It is $(L+1)$ instead of L due to the misalignment in the slots of the nodes i_1 and i_N in the unslotted 802.15.4 protocol.

To express the conditional probability in terms of ϕ, the transmission pattern needs to be understood: If there was no difference between sensing the channel and starting the transmission, then in the unslotted case no two nodes would be transmitting simultaneously since the probability that two

nodes start sensing simultaneously in the continuous case is zero. However, since there is a finite time between channel sensing and starting transmission, we assume that in the worst case, if two or more nodes start sensing in the same slot (slots are considered the same if the difference between their starting time is minimal), even if they are misaligned, the transmissions start at the same slot.

The conditional probability is hence equivalent to

$$\Pr\left(\bigcap_{k=2}^{n+1} S^{i_k}(s) = -1 \mid S^{i_1}(s) = -1, S^{i_N}(c) = -1\right) \simeq \Pr\left(\bigcap_{k=2}^{n+1} S^{i_k}(c) = -1 \mid S^{i_1}(c) = -1, S^{i_N}(c) = -1\right). \quad (4.23)$$

Since we assumed that the probability ϕ to sense in a given slot is independent across nodes, we can easily see that this is

$$\Pr\left(\bigcap_{k=2}^{n+1} S^{i_k}(c) = -1 \mid S^{i_1}(c) = -1, S^{i_N}(c) = -1\right) = \phi^n(1-\phi)^{N-2-n}, \quad (4.24)$$

which requires nodes $i_2, ..., i_{n+1}$ to sense and the remaining $N-2-n$ nodes not to sense in the sensing slot of i_1. As a result,

$$\alpha = (L+1)[1-(1-\phi)^{N-1}](1-\alpha). \quad (4.25)$$

From this, we can derive a second expression for ϕ:

$$\phi_2 = 1 - \left[1 - \frac{\alpha}{(L+1)(1-\alpha)}\right]^{\frac{1}{N-1}}.$$

The network operating point as determined by ϕ and α is given by solving the two non-linear Equations (4.20), (4.25).

We are now in the position to give the expression of the reliability. Recall that the reliability is defined as the probability of packet success. To have a successful packet transmission, the channel should be sensed idle when none of the other nodes is sensing the channel. The reliability is then given by the following simple expression:

$$R = \sum_{i=0}^{\mathrm{NB}} (1-\phi)^{N-1}(1-\alpha)\alpha^i = (1-\phi)^{N-1}(1-\alpha^{\mathrm{NB}+1}). \quad (4.26)$$

The average delay for the node, given that the packet is successfully transmitted, is given as follows:

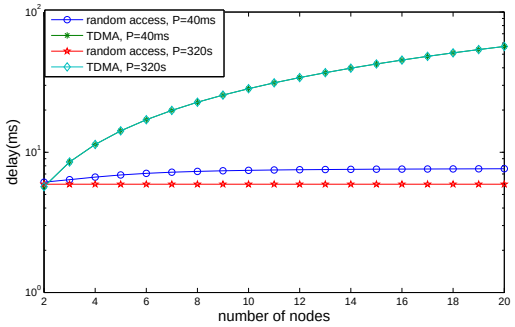


Figure 4.16 Delay for different number of nodes.

$$D = \left[\sum_{i=0}^{\mathrm{NB}} \left(\sum_{k=0}^{i} \frac{W_k + 1}{2} \right) \frac{\alpha^i (1-\alpha)}{1-\alpha^{\mathrm{NB}+1}} + L \right] r_s \,, \tag{4.27}$$

where r_s is the slot duration. Finally, the average energy consumption is given by the following expression:

$$E = P_l \left(\sum_{i=0}^{\mathrm{NB}} \sum_{k=1}^{W_i - 1} b_{i,k} + \sum_{i=0}^{\mathrm{NB}} b_{i,0} \right) + P_t \sum_{i=0}^{L-1} b_{-1,i} + P_s \sum_{i=0}^{X-1} b_{-2,i} \,. \tag{4.28}$$

where P_l, P_t and P_s are the average energy consumption in idle-listen, transmit and sleep states respectively. We assume that the radio is put in idle-listen state during the random backoff.

In the following, an analysis validation of the unslotted IEEE 802.15.4 and TDMA MAC protocol is presented [?]. For the implementation of the two MAC protocols, the software framework SPINE for health care applications, which runs on top of TinyOS 2.x (Gay et al., 2005) and Tmote Sky sensor nodes (sky, 2006) was used. In both the simulation and the experimental implementation we choose the default parameters for IEEE 802.15.4, namely $\mathrm{BE}_{\min} = 3$, $\mathrm{BE}_{\max} = 5$, $\mathrm{NB} = 4$, $f_s = 1/30Hz$ and $d = 40\mu s$ per second unless otherwise stated. Nodes were placed at few meters from the cluster-head, and in line of sight.

Figure 4.16 shows the delay as obtained by analysis and experiments for different number of nodes, whereas Figure 4.17 gives the delay for different packet generation periods. The delay increases considerably for TDMA systems as the number of nodes increases. On the other hand, the delay is almost constant for IEEE 802.15.4 especially at low packet generation rates. The curves achieved by our analysis match quite well the experimental behavior. The slight difference between analysis and simulations is due to the unavoidable measurements delay in the TinyOS protocol stack.

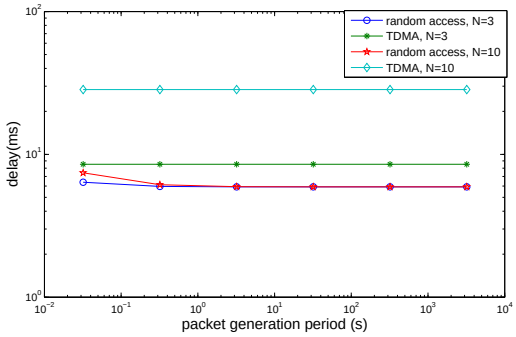


Figure 4.17 Delay for different number packet generation period.

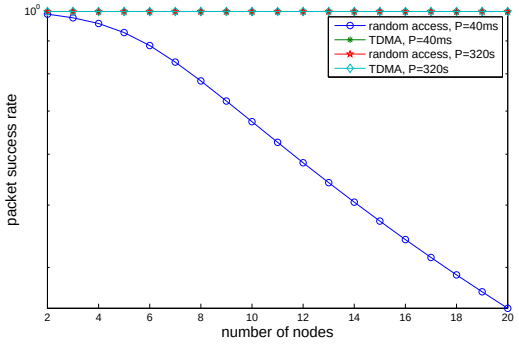


Figure 4.18 Reliability for different number of nodes.

Figure 4.18 shows the reliability for different number of nodes whereas Figure 4.19 gives the reliability for different packet generation periods as obtained by our analysis and experiments. The reliability is approximately 1 for TDMA MAC, except the small amount of packet losses in the experiments due to mainly rare channel attenuations and imperfect synchronization. The reliability is very close to 1 for random access schemes when the packet generation rate and number of nodes is low.

Figure 4.20 shows the energy for different number of nodes whereas Figure 4.21 gives the energy for different packet generation periods as obtained by analysis, cfr. (4.28) and (4.2), and experiments. The energy consumption of TDMA systems is better than random access protocols when the packet generation rate is very high and the number of nodes increases. However, as the packet generation rate decreases, the random access protocol performs better since there is no synchronization overhead. The small difference with the experimental results reflects the inaccuracies in the delay measurements, as we observed from the delay evaluation.

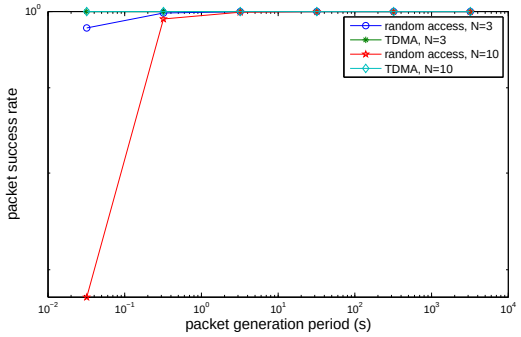


Figure 4.19 Reliability for different packet generation period.

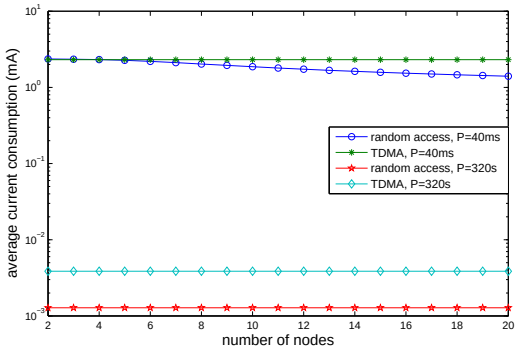


Figure 4.20 Average energy consumption for different number of nodes.

Data Transfer Model

There are three different approaches regarding the data transfer between a device and its coordinator. In the uplink mode, data is transferred to the coordinator by a particular device. On the other hand, in the downlink mode, data follows the opposite direction from the coordinator to the device. The last mode refers to the peer-to-peer mode where data is transferred between any peer network devices.

Uplink

In case of a beacon-enabled network, when a device wants to transmit data to a coordinator, it first listens for beacon frames. When a beacon frame is found, the device is synchronized with the superframe structure. During a particular time interval, the device transmits the data frame to the coordinator by using the CSMA/CA slotted mechanism. The coordinator accepts the successful frame inflow and it optionally transmits an acknowledgement frame which corresponds to the end of the transmission process.

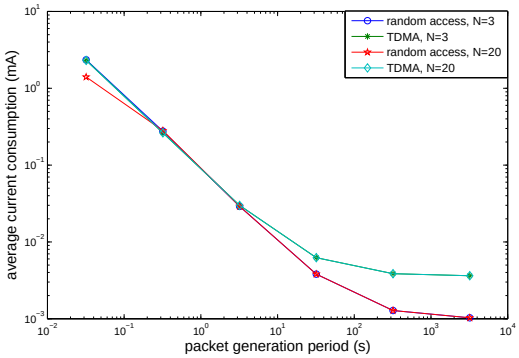


Figure 4.21 Average energy consumption for different packet generation period.

On the contrary, in case of a non beacon-enabled network, the device transmits the data frame to the coordinator by using the CSMA/CA unslotted mechanism. The coordinator optionally confirms the successful acceptance of the incoming frame by sending back an acknowledgement frame. These two different approaches are presented in Figure 4.22.

Downlink

In case of a beacon-enabled network, when a coordinator wants to transmit data to a particular device, it indicates to the beacon frame that a data message is pending. The device listens periodically for beacon frames and, if a data message is pending, it requests for the data by transmitting a MAC layer command with the use of CSMA/CA slotted mechanism. The coordinator confirms the request reception by sending back an acknowledgement frame. In addition, the data message that was pending is now sent with the use of CSMA/CA slotted mechanism. Finally, the device confirms the successfully incoming data frame by sending to the coordinator an acknowledgement frame. By the time the last acknowledgement frame is received by the coordinator, the data message is deleted from the list pending messages.

In case of a non beacon-enabled network, a coordinator, that wants to transmit data, keeps them until a device makes a request for them. A device creates a connection with the coordinator by transmitting a MAC layer command requesting for the data. In response to this request, the coordinator sends back an acknowledgement frame. In case of a pending data message, the coordinator sends the data frame using the CSMA/CA unslotted mechanism; otherwise, it sends a data frame with zero payload. The last step of this interconnection involves an acknowledgement frame sent back to the coordinator when the data frame is successfully received by the device. The above mentioned approaches are presented in Figure 4.23.

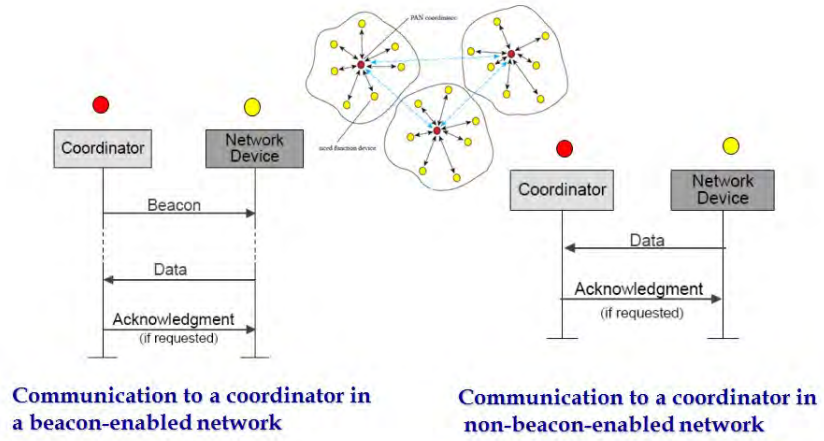


Figure 4.22 Uplink of a IEEE 802.15.4 WSN.

Peer-to-peer

In case of a peer-to-peer data transfer, every network device is able to communicate with another inside its communication range. If the devices can receive messages simultaneously, the transmission is performed with the use of the unslotted CSMA/CA mechanism; otherwise, specific measurements are needed in order to achieve synchronization between the devices.

Problems

Problem 4.1 Slotted Aloha

In this exercise we analyze the Slotted Aloha when the number of stations n is not exactly known. In each time slot each station transmits with probability p. The probability that the slot can be used (i.e. the probability that exactly one station transmits) is

$$\Pr(\text{success}) = n \cdot p(1-p)^{n-1} .$$

If n is fixed, we can maximize the above expression and get the optimal p. Now assume that the only thing we know about n is $A \leq n \leq B$, with A and B being two known constants.

(a) What is the value of p that maximizes Pr(success) for the worst $n \in [A, B]$?

(b) What is this "worst case optimal" value for p if $A = 100$ and $B = 200$?

Problem 4.2 Slotted ALOHA protocol

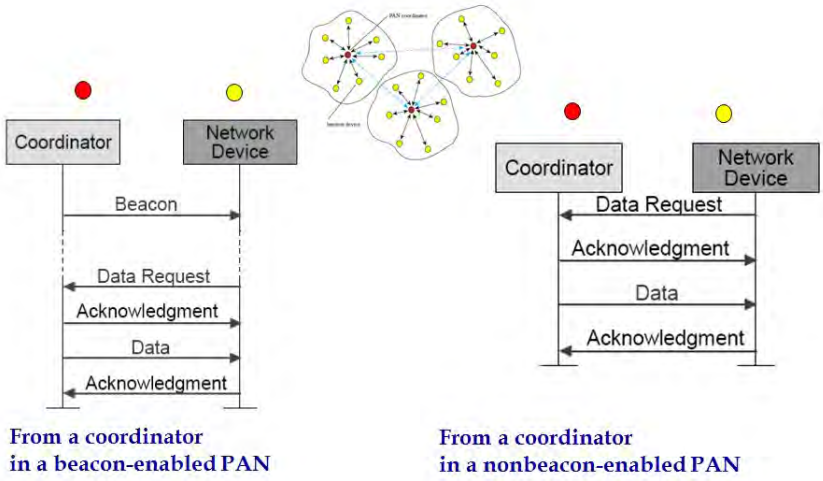


Figure 4.23 Downlink of an IEEE 802.15.4 WSN.

A dense WSN consists of 7200 sensors which are competing for the channel access according to a slotted ALOHA protocol. The capacity of each particular slot is equal to 4 Mbps. Each sensor node sends in average 25 transmission requests per hour each of them having a size of 80 byte. Compute the channel's total payload.

Problem 4.3 ARQ Ex.8.10 in (Pottie and Kaiser, 2005)

Consider a simple ARQ scheme through a single transmission link of data rate R. The ARQ scheme works as follows. The sender transmits a data packet across the link. Once the receiver receives the whole packet, it checks if data have been corrupted. If there is no error, a packet is sent to the sender to acknowledge the correct reception of the data packet. If there is an error, an ARQ is sent for a retransmission. The sender resends the packet immediately after it receives the ARQ packet. Assume the lengths of data and ARQ packets are L and L_{ARQ} respectively, and the propagation delay along the link is t_d . Neglect the turn-around time at the sender and the receiver. Suppose that the probability the data packet is corrupted during transmission is P_e and ARQ packets are always correctly received.

(a) Determine the average number of transmissions required for a packet to be correctly received.

(b) Find the average delay a packet experiences. The delay is defined as the time interval between the start of the first packet transmission and the end of the correct packet reception, and note that it does not include the transmission of the last acknowledgement packet.

Problem 4.4 Analysis of CSMA based MAC in WSNs

In this exercise we evaluate the performance of slotted CSMA protocol with fixed contention window size. Such mechanism is supported by protocols such as IEEE 802.15.4 in non-beacon enabled mode.

Assume a network of N sensors with a single channel and all the nodes are in the communication range of each other. The nodes use slotted CSMA scheme with fixed contention size M. Nodes sense the channel and if the channel is free they enter to the contention round. In contention round each node draws a random slot number in $[1, M]$ using uniform distribution and sets its counter with this integer number. In successive slots times t_{slot} each contender counts down until when its counter expires then it senses the channel and if there is no transmission in the channel it will send the packet immediately at beginning of the next slot. Assume t_d is the required time to transmit the data packet. t_{slot} is determined by physical layer parameters like propagation time of the packet (it also called vulnerable time) which is defined by the distance between the nodes. In this exercise t_{data} depends on data length is assumed to be much larger than t_{slot}. Each contention round will finish by a packet transmission that might be either successful or collided. Collision happens if at least two nodes draw the same minimum slot number, otherwise the transmission would be successful.

(a) Define P_s as the probability of having a successful transmission after a contention round with M maximum window size and N contenders. Also denote $p_s(m)$ as the probability of success at slot m. Find $p_s(m)$ and P_s.

(b) Similarly denote P_c as the probability of collision after contention round and $p_c(m)$ as the probability of collision at slot m. Propose an analytical model to calculate P_c and $p_c(m)$. Note that based on our system model, a collision happens at slot m, if at least two sensors pick the same slot m to transmit given that nobody has selected a smaller slot.

Problem 4.5 Malfunctioning Nodes and ARQ (Ex.14.7 in (Pottie and Kaiser, 2005))

A rectangular grid of sensor nodes is used for relaying packets. For the electronics used, it costs two times the energy of a hop among nearest neighboring nodes (separated by distance d) to hop diagonally across the square (e.g. node 2 to 5) and eight times the energy to go a distance of 2d in one hop (e.g. node 2 to 3). In normal operation, packet dropping rates are negligible and routes that use the least energy are chosen.

(a) Considering only energy consumption, at what packet dropping rate is it better to consider using two diagonal hops to move around a malfunctioning node?

(b) Now suppose delay constraints are such that we can only tolerate the probability of needing three transmission attempts being less than 0.01. In this case, what error rate is acceptable, assuming packing dropping events are independent?

Problem 4.6 MAC optimization for distributed estimation

Consider N nodes randomly deployed in a field. A node periodically checks with period S if there is an event of interest. Whenever node k detects such an event x_k, it starts broadcasting a monitoring message $m_k(x_k)$, which is called "state vector", to a fusion center. Nodes uses the slotted ALHOA medium access control protocol and transmit over the same wireless channel. In particular, each node transmits $m_k(x_k)$ in a randomly chosen time slot within the range $[1, S]$ units where S is the total number of slots per second. The node transmits a message within the slot boundaries following any slot. Hence, each node starts transmission with probability

$$\tau = \frac{z}{S}\,, \qquad 0 < \tau < 1\,,$$

where z is the rate of state vector transmissions per second. The probability that a node does not start transmission is $1 - \tau$. Collision at the fusion center happens when two nodes simultaneously transmit in a time slot.

The state vector transmission interval is $T_u = 1/z$. Each node wants to minimize the state vector transmission interval so to have often and more reliable information about the event of interest. However, this increases the collision probability.

(a) Pose an optimization problem which copes with such a tradeoff and argue if it is a convex one.

(b) Calculate the optimal rate of state vector transmissions per second that minimizes T_u.

Problem 4.7 Broadcast

Three students discuss the broadcasting problem with collision detection in graphs of constant diameter. Student A claims that there is a deterministic protocol that allows to broadcast messages of length l in time $O(l)$. He says that it is possible since all nodes act synchronously and can detect collisions, which allows to transmit information one bit per round (slot) using the collision detection mechanism, i.e. detecting a transmission or collision in a slot means bit 1, detecting a free channel means 0. Student B says that this is not possible because he can prove the existence of a lower bound of $\Omega(\log n)$ for deterministic algorithms, which can be much larger than the length of a message l in general. He says that this can be done in the same way as for the lower bound of n for the deterministic broadcast without collision detection for graphs of diameter 2, i.e. using golden and blue nodes in the middle layer. Student C claims that A's idea works in principle but all nodes need to know the length l of the message. Who is right?

(a) If you believe A is right, give an algorithm that performs the broadcast.

(b) If you believe B is right, give a proof.

(c) If you believe C is right, describe an algorithm given that all nodes know the message length l and explain why the message length l is needed.

Problem 4.8 MAC with NACK

Suppose that a sensor node A transmits packets to the sink node S. Across this link, 10 sensor nodes $N_1, N_2, ..., N_{10}$ forward the packets acting as relays. The total packet delay in each relay node is 100 ms. However, in average 1 out of 10 packets that reach to S have problems and need to be retransmitted. In this case, S sends a Negative Acknowledgment (NACK) to A in which explicitly notifies the sender which packets need to be retransmitted. The NACK packets reach in A after 300 ms. In addition, in order to avoid the existence of stale copies of packets within the WSN, the relay nodes implement an elimination policy of old packets with an age larger than T. The parameter T is chosen equal to twice the average time a packet needs to reach from A to S.

(a) Compute and comment the value of parameter T.

(b) What is the percentage of packets that are eliminated before reaching S?

Problem 4.9 $M/M/1$ queues (Ex.8.9 in (Pottie and Kaiser, 2005))

Consider the infinite length $M/M/1$ queue.

(a) Given that the probability of n customers in the queue is $p(n) = (1-\rho)\rho^n$, where $\rho = \lambda/\mu$, show that the average number of customers in the queue is

$$N = \mathbb{E}(n) = \sum_{n=0}^{\infty} np(n) = \frac{\rho}{1-\rho}.$$

(b) Plot N as a function of ρ when $0 \leq \rho < 1$. What happens when $\rho \geq 1$?

(c) Find the average delay that customers experience and the average waiting time that customers spend in queue. (Hint: use Little's theorem.)

Chapter 5

Routing

5.1 Introduction

In the previous chapter, we studied the channel access control mechanisms. The mechanism by which nodes get the right to transmit in a shared communication medium is of crucial importance; the minimization of the resulting collisions, the fairness among the nodes, and the overall energy efficiency are major concerns in the design of MAC protocols in WSNs. The next essential question that arises is on which path messages should be routed when multiple patch are available from a source node to a destination node? Moreover, which are the basic routing options and how to compute the shortest routing path along which messages should be sent? In this chapter, fundamental aspects concerning the routing mechanisms are examined.

Routing is formally defined as the mechanism of determining a path between the source and the destination node upon request of message transmission from a given node. In WSNs, the network layer is mostly used to implement the routing of data messages. In case of large multi-hop networks, the source node cannot reach the destination directly, and, therefore, intermediate nodes have to relay their messages. An intermediate node has to decide to which neighbor an incoming message should be forwarded if the message is not destined to itself.

Traditionally, routing tables that list the most appropriate neighbor for any given message destination are used. The implementation of routing tables concerning a particular routing algorithm provides the paths for each destination. The construction and maintenance of these routing tables is the crucial task of both a centralized and a distributed routing protocol in WSNs. The building of these tables basically reduced to establishing what is the path from a given node to reach a given destination. How this is done, is the focus of this chapter.

The chapter is organized as follows: In Section 5.2, an overview of the

main challenges that occur in the routing mechanism are reviewed. In Section 5.3, a classification of routing protocols is made, based on various criteria, and the most representative protocols are presented. Next, the shortest path optimization problem is summarized accompanied with a generic algorithm that is used to solve it. After covering the most significant routing metrics, the final section of the chapter provides a brief description of the ROLL RPL routing protocol for WSNs.

5.2 Routing Challenges

Routing in WSNs is very challenging due to the inherent characteristics that distinguish these networks from other wireless networks, such as mobile ad hoc networks or cellular networks. Due to the unique characteristics and peculiarities of a WSN, the existing routing protocols developed for wireless ad hoc networks cannot be directly applied to WSNs. The design of a routing protocol for WSNs has to take account the unreliability of the wireless channel, the potential dynamic changes in the network topology, as well as the limited processing, storage, bandwidth, and energy capacities of the WSN nodes. Therefore, special approaches are required to ensure efficient routing amongst the nodes of a WSN.

The design of routing protocols in WSNs is influenced by many factors. These factors poses several challenges that must be overcome before efficient communication can be achieved in WSNs. In the following, we summarize some of the routing challenges and design issues that affect routing in WSNs.

- *Energy consumption*: Energy consumption is considered one of the major concerns in the development routing protocols for WSNs. Sensor nodes can drain their limited supply of energy while performing computations and transmitting information in a wireless environment. While building their routing tables, nodes consume energy by exchanging information with their neighbors. Furthermore, because of the limited energy resources of WSN nodes, messages need to be delivered in the most energy-efficient manner without compromising the accuracy of the information content. Shortest path algorithms need to adopt metrics such as energy-efficiency.

- *Scalability*: As the size of the network increases, or the number of nodes increases, the routing protocol should be able to adapt to the changes and provide adequate performance. WSNs may consist of a large number of nodes, and therefore, the information each node obtains about the network topology is limited. Hence, fully distributed protocols, which operate with limited knowledge of the topology, need to be developed to provide scalability. In addition, when the density

may be high in the network, local information exchange should also be limited to improve the energy efficiency.

- *Node mobility*: In some cases, nodes are not stationary. Node mobility introduces changes in the neighborhood relations; nodes move out of range of their neighbors and hence are no longer able to communicate with the old neighboring nodes while they come within the range of new nodes. An ideal routing protocol for WSN should be able to deliver data messages from source to destination even when some of the intermediate nodes move away from their neighbors range. This complicates the design of the routing protocol as it introduces additional routing overhead. Route stability is an important issue, in addition to energy and other aspects. Hence, the routing protocol should be adaptive to dynamic changes in the network topology.

- *Node deployment*: Node deployment in WSNs can be either deterministic or randomly performed, which is dependent on the required application. In the deterministic approach to deployment, all the nodes are placed in predefined positions and messages are routed through paths that are pre-determined. However, in the randomly deployed nodes, nodes may be placed randomly in arbitrary positions. The network topology can change dynamically during the lifetime of the network. Initially, sensor nodes may be unaware of the network topology. The relative locations of the neighbors of a node and the relative location of the nodes in the network significantly affect the routing performance. This is exactly a task of a routing protocol which should provide topology information such that the neighbors of each node are discovered and routing decisions are made accordingly.

- *Robustness*: Routing in WSNs is based on the sensor nodes to deliver data in a multi-hop manner. Hence, routing protocols operate on these nodes instead of dedicated routers such as in the Internet. These nodes consist of low cost hardware which may result in unexpected failures to such an extent that a node may be non-operational. As a result, routing protocols should provide robustness to sensor node failures.

- *Application*: The type of application is also important for the design of routing protocols. In case of monitoring applications, static routes can be reused to maintain efficient delivery of the observations throughout the lifetime of the network. On the other hand, in event-based applications, since the nodes are usually in sleep mode, whenever an event occurs, routes should be generated to deliver the event information in a timely manner.

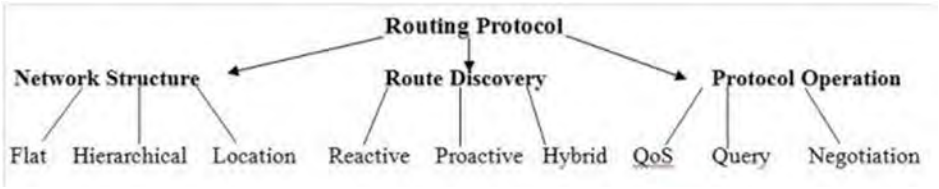


Figure 5.1 WSN Routing Protocol classification.

5.3 Routing Protocols Classification

Routing protocols for WSNs can be classified in different ways depending on various criteria. In Figure 5.1, a general classification is reported. In the following sections, the classification is explained in the detail.

5.3.1 Network Structure

The underlying network structure can play significant role in the operation of the routing protocol in WSNs. With respect to the network organization, most routing protocols fit into one of three classes; flat, hierarchical and location-based protocols. In this section, we survey the protocols that fall in this category.

Flat and Data Centric Protocols

In flat-based (Data Centric) routing protocols all the nodes are considered equal with respect to their role and functionality. Due to the potentially large number of nodes deployed in many applications of WSNs, it is not feasible to assign global identifiers to each node. Therefore, routing protocols that are based on unique identifiers (IDs) or addresses are not suitable for WSNs. To provide the solution for this problem, data centric routing protocols have been proposed. In these routing techniques, the focus is on the retrieval and dissemination of information of a particular type or described by certain attributes, as opposed to the data collection from particular nodes.

Flooding, the WSN protocols for Information via Negotiation (SPIN) and directed diffusion are few examples of routing protocols that may apply data-centric techniques. In the following sections, we provide an overview of some of these data-centric based routing protocols in flat-based networks.

- **Flooding**
  A simple strategy to disseminate information into a network is to flood the entire network (Figure 5.2). The sender node broadcasts packets to its immediate neighbors, which will repeat this process by rebroadcasting the packets to their own neighbors until all nodes have received the packets or the packets have travelled for a maximum number of hops.

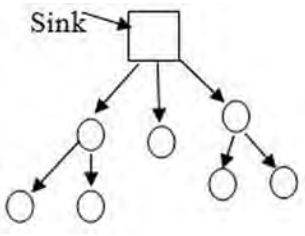


Figure 5.2 Flooding mechanism of a routing protocol.

With flooding, if there exists a path to the destination (and assuming lossless communication), the destination is guaranteed to receive the data.

Flooding is very simple to deploy without any algorithms for route discovery. It is also robust with respect to changes in the network topology. The main advantage of flooding is the increased reliability provided by this routing method. Since the message will be sent to at least once to every host it is almost guaranteed to reach its destination. On the other hand, the drawback is that heavy traffic is generated throughout the network without considering the resource constraints of individual nodes. In some cases, a node receives duplicated packets from its neighbors leading to a waste of the bandwidth. Therefore, measures are taken in order to ensure that packets do not travel through the network indefinitely. For example, maximum-hop counts are used to limit the number of times a packet is forwarded. The hop counter is usually contained in the header of each packet and is decremented at each hop, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet.

Its value should be set large enough so that every intended receiver can be reached, but also small enough to ensure that packets do not travel too long in the network. An alternative technique for damming the flood is to keep track of which packets have been flooded in order to avoid sending them a second time. For this reason, sequence numbers in packets (combined with the address of the source) can be used to uniquely identify a packet. When a node receives a packet that it has already forwarded (i.e., with the same destination–source pair and the same sequence number), it simply discards this duplicate.

- **SPIN**
  Sensor Protocols for Information via Negotiation (SPIN) (Kulik et al.

2002) is a family of negotiation-based, data-centric, and time-driven flooding protocols. However, compared to classic flooding, SPIN nodes rely on two key techniques to overcome the deficiencies of flooding. SPIN nodes negotiate with their neighbors before they transmit data, allowing them to avoid unnecessary communications and each SPIN node uses a resource manager to keep track of actual resource consumption, allowing an adaptive routing and communication behavior based on resource availability.

SPIN is a combination of different protocols including SPIN-PP, SPIN-EC, SPIN-BC, and SPIN-RL. SPIN-PP is best suited for point-to-point traffic and provides optimized routing solution when nodes communicate directly without interference from other nodes. Data transmission in SPIN-PP consists of three steps; in the first step a node advertises the data to its neighbors by using advertisement message (ADV). Neighbors nodes perform some check when they receive the ADV message to verify that they already have the described data or not. If neighbor nodes do not have the described data then they request the data by sending REQ message. Finally, send the data by sending DATA message that contain the advertised data. SPIN-EC is a second member of SPIN group and it works in the same way as SPIN-PP but with the addition of energy conservation. In SPIN-EC nodes only participate in three-way handshake if they have enough energy resource level to complete the process of data transmission. Unlike first two member types of SPIN, SPIN-BC support broadcast transmission where every node in a network will receive the transmitted message. SPIN-BC adopts the one-to-many data transmission model where every node hears all the transactions within its range. The last member is SPIN-RL that also broadcasts the traffic, provides a mechanism for detection of packet loss and also addresses the asymmetric communications.

- **Directed diffusion**

  Directed diffusion is another data centric protocol. The data is named using attribute-value pairs and it is the collected or processed information of a phenomenon that matches an interest of a user. The main idea of directed diffusion is that nodes request data by sending interests for named data which are then flooded over the whole network. Whenever a node receives an interest, it will check whether the interest exists or new one. If it is a new interest, the sensor node will set up a gradient toward the sender to draw down data that matches the interest. Each pair of neighboring nodes will establish a gradient to each other. After the gradient establishment stage, the source node begins to send the related data that matches the interest. While sensor data is transferred

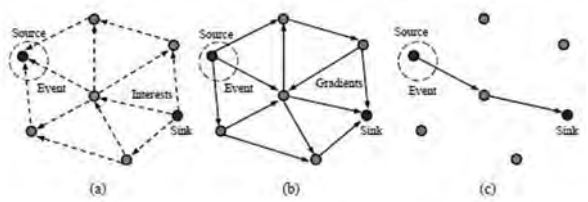


Figure 5.3 Directed diffusion routing: (a) Interest propagation (b) Initial gradient setup (c) Data delivery.

to the recipient in response to the interest, intermediate nodes combine data from different sources to eliminate redundancy and reduce the number of transmissions. This process is shown in Figure 5.3.

Directed diffusion differs from SPIN in that queries (interests) are issued on demand by the sinks and not advertised by the sources as in SPIN. Based on the process of establishing gradients, all communication is neighbor-to-neighbor, removing the need for addressing schemes and allowing each node to perform aggregation and caching of WSN data, both of which can contribute to reduced energy consumption. Finally, directed diffusion is a query-based protocol, which may not be a good choice for certain sensor network applications, particularly where continuous data transfers are required (e.g., environmental monitoring applications).

Hierarchical Protocols

Hierarchical routing protocols are based on the grouping of nodes into clusters to address some weaknesses of flat routing protocols, most notably scalability and efficiency. In this particular technique, sensor nodes are grouped into clusters where all the nodes communicate only directly with the leader node (cluster head) within their own cluster as shown in the Figure 5.4. These leader nodes have more power and less energy constraints and are responsible with the forwarding of the messages on behalf of the other nodes. This approach can significantly reduce the communication and energy burdens on sensor nodes, while cluster heads will experience significantly more traffic than regular sensor nodes. Challenges in the design and operation of hierarchical routing protocols include the selection of cluster heads, the formation of clusters, and adaptations to network dynamics such as mobility or cluster head failures. Compared to flat routing approaches, hierarchical solutions may reduce collisions in the wireless medium and facilitate the duty cycling of sensor nodes for increased energy efficiency.

Low Energy Adaptive Clustering Hierarchy (LEACH) is an adaptive clustering and self organizing protocol. LEACH assumes that every cluster head

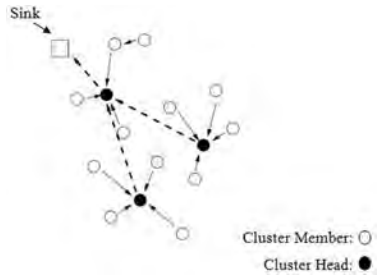


Figure 5.4 Hierarchical architecture a WSNs.

can directly communicate with the base station.With LEACH, cluster heads are responsible for all communication between their cluster members and a base station and the aggregation of data coming from its cluster members in order to eliminate redundancies. LEACH can achieve significant energy savings (depending on how much redundancy can be removed) and WSN nodes (apart from the cluster heads) are not responsible for forwarding data of other nodes.

Location-based Protocols

Location-based also known as geographic routing protocols rely on the location information from nodes instead of topological connectivity information to make routing decisions. In unicast location-based routing, packets are sent directly to a single destination, which is identified by its location. That is, a sender must be aware not only of its own location, but also the location of the destination. In broadcast or multicast location-based routing approaches, the same packet must be disseminated to multiple destinations. Multicast protocols take advantage of the known destination locations to minimize resource consumption by reducing redundant links.

Location based technique is an optimized solution for routing where WSN nodes are not required to establish paths between source and destination and also do not need to maintain routing tables. Typically, location-based routing protocols require that every node in the network knows its own geographic location and the identities and locations of its one-hop neighbors (e.g.,obtained via periodic beacon messages). The destination is expressed either as the location of a node (instead of a unique address) or a geographic region.

Greedy Perimeter Stateless Routing (GPSR) is a location based routing protocol in which packet forwarding decision are based on node position and destination. In GPSR, nodes obtain information about its directly connected nodes via HELLO or beacon messages. Source node mark the sending data with the location of the receiving node and relay a packet to its immediate

neighbor that make a locally forwarding decision and handle the data to neighbor that is geographically close to the destination. Every node from source to destination make local forwarding decision and move data closer to the destination hop by hop, until the destination is reached.

Compared to other routing solutions, an advantage of location-based routing is that only geographic information is needed for forwarding decisions and it is not necessary to maintain routing tables or to establish end-to-end paths between sources and destinations, eliminating the need for control packets (apart from the beacon messages among neighbors).

5.3.2 Route Discovery

As mentioned before, routing protocols are responsible for identifying and selecting routes from a source to a destination node. This route discovery process can also be used to distinguish between different types of routing protocols. Reactive protocols discover routes on-demand, that is, route is only determined when needed by a node. An example of an on-demand or reactive protocol is the Ad Hoc On-Demand Distance Vector (AODV) protocol. AODV relies on a broadcast route discovery mechanism, which is used to dynamically establish route table entries at intermediate nodes. Whenever a node needs to send a message to a node that is not its neighbor, it initiates a path discovery process, by broadcasting a Route REQuest (RREQ) message to its neighbors. Nodes receiving the RREQ update their information about the source. They also set up a backward link to the source in their routing tables. Each RREQ message contains the addresses of the source and the destination, a hop count value, a broadcast ID, and two sequence numbers. The hop count value keeps track of the number of hops from the source while the broadcast ID is incremented whenever the source issues a new RREQ packet and is combined with the source's address to uniquely identify an RREQ.

Upon receiving an RREQ packet, a node that possesses a current route to the specified destination responds by sending a unicast route reply (RREP) message directly back to the neighbor from which the RREQ was received. Otherwise the RREQ is rebroadcast to the intermediate node's neighbors and its hop count is increased by one. It should be noted that intermediate nodes reply to an RREQ only if the sequence number of their route to the destination is greater than or equal to the destination sequence number specified in the RREQ packet.

When the source node receives the RREP, it checks whether it has an entry for the route. If it did not have any entry in its routing table, the node creates a new entry in the routing table. Otherwise it checks the sequence number of the RREP. If the RREP arrives with the same sequence number as in its tables but with a smaller hop count, or a greater sequence number (indicating fresher route), it updates its routing table and starts using this

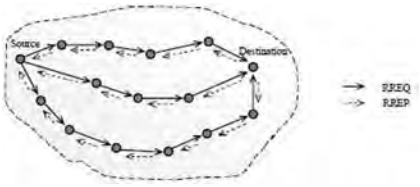


Figure 5.5 Path discovery of the AODV routing protocol.

better route. Once an entry for the new route has been created in the table, the node can start communication with the destination. Figure 5.5 summarizes the path discovery procedure of the AODV protocol.

The Dynamic Source Routing (DSR) protocol employs route discovery and route maintenance procedures similar to AODV. In DSR, each node maintains a route cache with entries that are continuously updated as a node learns new routes. Similar to AODV, a node wishing to send a packet will first inspect its route cache to see whether it already has a route to the destination. If there is no valid route in the cache, the sender initiates a route discovery procedure by broadcasting a route request packet, which contains the address of the destination, the address of the source, and a unique request ID. As this request propagates through the network, each node inserts its own address into the request packet before rebroadcasting it. As a consequence, a request packet records a route consisting of all nodes it has visited. Unlike AODV, each packet in DSR carries route information, which allows intermediate nodes to add new routes proactively to their own caches. Also, DSR's support of asymmetric links is another advantage compared to AODV.

On the other hand, in proactive protocols, the routing tables are kept constantly up-to-date and active before they are actually needed. In this way, the delays before the actual data transmission are eliminated. However, unnecessary routes may be established and also the time interval between the route discovery and the actual use of the route can be very large, potentially leading to stale information in the routing tables which leads to routing errors. In addition, another drawback refers to the overheads involved in building and maintaining potentially very large routing tables. A typical proactive protocol is the Destination-Sequenced Distance Vector (DSDV) routing protocol (Perkins and Bhagwat 1994) which is a modified version of the classic Distributed Bellman-Ford algorithm. Each node maintains a vector (table) of minimum distance to every node via each of his neighbors and broadcasts updates to the routing table periodically, but also immediately whenever significant new information becomes available. This information is stored in a routing table, along with a sequence number for each entry, where this number is assigned by the destination node. The purpose of the

sequence numbers is to allow nodes to distinguish stale routes from new ones in order to prevent routing loops.

Another example of a proactive protocol is the Optimized Link State Routing (OLSR) protocol (Clausen et al. 2001), which is based on the link state algorithm. In this approach, nodes periodically broadcast topological information updates to all other nodes in the network, allowing them to obtain a complete topological map of the network and to immediately determine paths to any destination in the network.

Some protocols exhibit characteristics of both reactive and proactive protocols and belong to the category of hybrid routing protocols.

5.3.3 Protocol Operation

Another major classification of routing protocols refers to their operation.

In the QoS-based routing protocols, the goal is to find feasible paths between sender and destination, while satisfying one or more QoS metrics (latency, energy, bandwidth, reliability), but also optimizing the use of the scarce network resources. In some WSN applications, Quality-of-Service related metrics like delay, jitter, and throughput have same importance as energy consumption. Wireless sensor networks pose numerous challenges for providing satisfactory QoS, including dynamic topologies, resource scarcity (including power limitations), varying quality of the radio channels, the lack of centralized control, and the heterogeneity of network devices.

Sequential Assignment Routing (SAR) is a multipath routing approach that explicitly considers the QoS metrics. SAR creates multiple trees that are rooted from one hope neighbor nodes of the sink. Trees grow outward from a sink while avoiding those nodes that have low QoS and energy. SAR selection for route is based on QoS metric, packet priority level, and energy. Every node in a network is a part of multiple trees and it can select multiple routes toward the sink. Multiple paths availability provides fault tolerance and fast recovery from broken paths. In large networks, multiple trees establishment and maintaining are expensive.

SPEED is a QoS based routing protocol that provides real time unicast, multicast, and anycast communication services. Nodes in SPEED protocol relies on location information of sensor nodes instead of routing tables, therefore SPEED is also a location based stateless routing protocol. SPEED protocol based on several different components to make sure speed guarantees to the packets. SPEED is scalable and efficient protocol suited for highly dense environments where large numbers of nodes with limited resources are deployed.

Query-based routing protocols are initiated by the receiver node. Specifically, in this kind of routing, the destination nodes propagate a query for data (sensing task) from a node through the network and a node having this data sends the data which matches the query back to the node, which

initiates the query. All the nodes have tables consisting of the sensing tasks queries that they receive and send data which matches these tasks when they receive it. Directed diffusion which was described in Section 5.3.1 is an example of this type of routing. In directed diffusion, the BS node sends out interest messages to sensors. As the interest is propagated throughout the sensor network, the gradients from the source back to the BS are set up. When the source has data for the interest, the source sends the data along the interests gradient path.

Negotiation-based protocols aim to reduce redundant data transmissions. The main idea of negotiation based routing in WSNs is to suppress duplicate information and prevent redundant data from being sent to the next sensor or the base-station by conducting a series of negotiation messages before the real data transmission begins. Consequently, less energy is consumed since duplicate information is avoided. The SPIN family protocols discussed earlier is an example of negotiation-based routing protocols.

5.3.4 In-network Data Processing

Finally, routing protocols also differ in the way they support in-network data processing.

In non coherent-based protocols, nodes may perform significant data processing before it is forwarded to other nodes for further processing. The nodes that perform further processing are called the aggregators.

On the other hand, coherent-based protocols perform only a minimum amount of processing (e.g., eliminating duplicates, time-stamping) before sensor data is sent to receivers and data aggregators. The minimum processing typically includes tasks like time stamping, duplicate suppression, etc. When all nodes are sources and send their data to the central aggregator node, a large amount of energy will be consumed and hence this process has a high cost. One way to lower the energy cost is to limit the number of sources that can send data to the central aggregator node.

5.4 The Shortest Path Routing

Routing protocols decision about best route selection among multiple routes is based on the type of route metric that is used. The main concepts that are related with the choice of a particular routing option refer to the delay, the energy (maximum battery capacity/ minimum battery cost) and the packet error rate. However, independently with the chosen routing metric, there is a basic way to model all these options. The shortest path algorithm computes the path for each source-destination pair that leads to the least cost (energy, latency). Many fast-converging techniques exist for the problem of finding the shortest routes. The shortest-path algorithm performs well in cases of slow network changes (so that the shortest routes do not

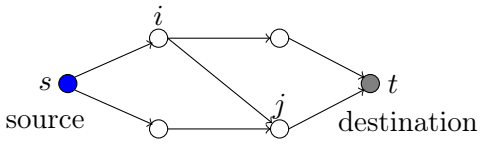


Figure 5.6 Basic version of shortest path optimization problem for routing over networks.

have to be continually rediscovered), and mild loading. On the other hand, in cases of network congestion, the algorithm may not actually be optimal as too many packets may be dropped. This section presents the shortest path optimization problem focusing also on the iterative generic shortest path algorithm for the optimal path computation in a WSN. In the end of the section, some of the most common used routing metrics based on the shortest path problem are presented

5.4.1 The Shortest Path Optimization Problem

The shortest path optimization problem is a general optimization problem that is used to model all the existing cases for routing (used in ROLL RPL, Wireless HART...). The broad range of applications of the shortest path problem covers project management topics, dynamic programming as well as the paragraphing problem. In this section, its basic version is examined, when in the network there is one source and one destination (Figure 5.6). Multiple sources or multiple destinations scenarios are a simple extension. In general, a graph is used to formulate routing problems. A graph $G = (N, A)$ is a set N of nodes and a collection $A = (i, j)$ of edges/arcs, where each arc is a pair of nodes from N. In the context of this section, every arc length a_{ij} is a scalar number that represents the chosen routing cost on the link between node i and node j (MAC delay, Packet Error Rate...). Based on Figure 5.6, suppose that we want to find the shortest (minimum cost) path from source s to destination t. The shortest path optimization problem is then formulated as follows:

$$\min_{\mathbf{x}} \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij}$$

$$\text{s.t.} \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji} = s_i \begin{cases} 1 & \text{if i=s} \\ -1 & \text{if i=t} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A}$$

$$\mathbf{x} = [x_{12}, x_{13}, ..., x_{i_n}, x_{i_{n+1}}, ...],$$

where x_{ij} represents a binary variable. It can be also real, but remarkably if the optimization problem is feasible, the unique optimal solution is binary. The optimal solution gives the shortest path between source s and destination t. Several approaches have been proposed for the solution of the shortest path problem. Since it is an optimization problem, one could use standard techniques of optimization theory, such as Lagrangian methods. However, the solution can be achieved by combinatorial algorithms that do not use optimization theory at all. Such a combinatorial solution algorithm, referred as the Generic shortest path algorithm, is presented in the following section.

5.4.2 The Generic Shortest Path Algorithm

The Generic shortest path algorithm is the foundation of other more advanced algorithms widely used for routing (e.g., in ROLL RPL) such as the Bellman-Ford and Dijkstra method. The algorithm maintains and adjusts a vector $(d_1, d_2, ..., d_N)$, where each d_j, called the label of node j is either a scalar or ∞. The use of labels is motivated by the Complementary Slackness (CS) conditions which are given in the following proposition.

Proposition 1. *Let $d_1, d_2, ..., d_N$ be scalars such that*

$$d_j \leq d_i + a_{ij}, \quad \forall (i,j) \in \mathcal{A}\,. \tag{5.1}$$

Let P be a path starting at a node i_1 and ending at a node i_k. If

$$d_j = d_i + a_{ij}, \quad \forall (i,j)\ of \quad P \tag{5.2}$$

P is a shortest path from i_1 to i_k.

The CS conditions (5.1) and (5.2) are considered as the foundation of the generic shortest path algorithm. In particular, after assigning an initial vector of labels $(d_1, d_2, ..., d_N)$ to the nodes, the arcs (i,j) that violate the CS condition $d_j > d_i + a_{ij}$ are selected and their labels redefined so that $d_j := d_i + a_{ij}$. In other words, if $d_j > d_i + a_{ij}$ for some arc (i,j), the path obtained by extending the path P_i by arc (i,j), which has length $d_i + a_{ij}$, is a better path (lower cost) than the current path P_j, which has length d_j. This redefinition is continued until the CS condition $d_j \leq d_i + a_{ij}$ is satisfied for all arcs (i,j). In this way, the algorithm finds successively better paths form the origin to various destinations.

Instead of examining arbitrarily the network nodes, the generic shortest path algorithm maintains a list of nodes V which is called the candidate list. In each algorithm iteration, the violation of the CS condition is checked for all the outgoing arcs of each particular node of the set V and the vector of labels $(d_1, d_2, ..., d_N)$ is updated. Assuming that initially $V = \{1\}$ and

$d_1 = 0,\ \ d_i = \infty,\ \ \forall i \neq 1$, the typical iteration (assuming V is non-empty) is as follows:

Iteration of the Generic Shortest Path Algorithm

Remove a node i from the candidate list V. For each outgoing arc $(i, j) \in \mathcal{A}$, if $d_j > d_i + a_{ij}$, set

$$d_j := d_i + a_{ij}\ ,$$

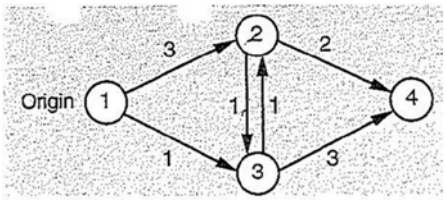
and add j to V if it does not already belong to V

The removal rule is based either on global routing algorithms or on decentralized ones.

A global routing algorithm computes the least-cost path between a source and destination using complete, global knowledge about the network. That is, the algorithm takes the connectivity between all nodes and all link costs as inputs. This then requires that the algorithm somehow obtain this information before actually performing the calculation. The calculation itself can be run at one site (a centralized global routing algorithm) or replicated at multiple sites. The key distinguishing feature here, however, is that a global algorithm has complete information about connectivity and link costs. In practice, algorithms with global state information are often referred to as link-state (LS) algorithms, since the algorithm must be aware of the cost of each link in the network. A representative example is the Dijkstra method. Dijkstra's algorithm computes the least-cost path from one node (the source) to all other nodes in the network. It is an iterative method and has the property that after the k-th iteration, the least-cost paths are known to k destination nodes, and among the least-cost paths to all destination nodes, these k paths will have the k smallest costs.

On the other hand, in a decentralized routing algorithm (often referred as distance-vector algorithm), the calculation of the least-cost path is carried out in an iterative, distributed manner. No node has complete information about the costs of all network links. Instead, each node begins with only the knowledge of the costs of its own directly attached links. Then, through an iterative process of calculation and exchange of information with its neighboring nodes (that is, nodes that are at the other end of links to which it itself is attached), a node gradually calculates the least-cost path to a destination or set of destinations. Bellman-Ford method is representative example of a decentralized routing algorithm. The basic idea is as follows. Each node x begins with $D_x(y)$, an estimate of the cost of the least-cost path from itself to node y, for all nodes in N. Let $D_x = [D_x(y) : y\ \ in\ \ N]$ be node x's distance vector, which is the vector of cost estimates from x to all other nodes, y, in N. With the DV algorithm, each node x maintains the following routing information:

- For each neighbor v, the cost $c(x, v)$ from x to directly attached neighbor, v



Iteration	Candidate List V	Node Labels	Node out of V
1	{1}	(0,∞,∞,∞)	1
2	{2, 3}	(0,3,1,∞)	2
3	{3, 4}	(0,3,1,5)	3
4	{4, 2}	(0,2,1,4)	4
5	{2}	(0,2,1,4)	2
	∅	(0,2,1,4)	

Figure 5.7 Illustration of the generic shortest path algorithm, from [Bertsekas 1991].

- Node x's distance vector, that is, $D_x = [D_x(y) : y \ in \ N]$, containing xâĂŹs estimate of its cost to all destinations, y, in N
- The distance vectors of each of its neighbors, that is, $D_v = [D_v(y) : y \ in \ N]$ for each neighbor v of x.

Each node sends a copy of its distance vector to each of its neighbors. When a node x receives a new distance vector from any of its neighbors v, it saves v's distance vector, and then uses the Bellman-Ford equation to update its own distance vector as follows:

$\mathrm{D}_x(\mathrm{y}) = \min_v\{\mathrm{c}(\mathrm{x},\mathrm{v}) + \mathrm{D}_v(\mathrm{y})\}$ for each node y in N

If node x's distance vector has changed as a result of this update step, node x will then send its updated distance vector to each of its neighbors, which can in turn update their own distance vectors. Miraculously enough, as long as all the nodes continue to exchange their distance vectors in an asynchronous fashion, each cost estimate $D_x(y)$ converges to $d_x(y)$, the actual cost of the least-cost path from node x to node y [Bertsekas 1991].

An illustration of the generic shortest path algorithm is shown in Figure 5.7. The numbers next to the arcs represent the arc lengths. It should be noted that the order in which the nodes are removed from the candidate list V is significant; if node 3 (instead of node 2) had been removed in iteration step 2, each node would enter V only at once. As the iterations of the algorithm increase, the vector of the labels takes monotonically non-increasing values. The following proposition refers to the convergence of the algorithm.

These convergence properties are based on sound theoretical analysis.

Proposition 2. *Consider the generic shortest path algorithm.*

(a) At the end of each iteration, the following conditions hold:

- *If $d_j < \infty$, then d_j is the length of some path that starts at 1 and ends at j.*
- *If $i \notin V$, then either $d_i = \infty$ or else*

$$d_j \leq d_i + a_{ij}, \quad \forall j \text{ such that } (i,j) \in \mathcal{A} \ ;$$

(b) If the algorithm terminates, then upon termination, for all j with $d_j < \infty, d_j$ is the shortest distance from 1 to j and

$$d_j = \begin{cases} \min_{(i,j)\in\mathcal{A}}(d_i + a_{ij}) & \text{if } j \neq 1 \\ 0 & \text{if } j = 1 \end{cases} \ ;$$

(c) If the algorithm does not terminate, then there exists some node j and a sequence of paths that start at 1, ends at j, and have a length diverging to $-\infty$.

(d) The algorithm terminates if and only if there is no path that starts at 1 and contains a cycle with negative length.

5.4.3 Routing Metrics

Based on the concept of the shortest path routing, the most common used metrics for routing in WSNs are discussed in the following section. Many other routing metrics besides the hop count and the energy are possible. For example, each arc could be labeled with the mean transmission delay for some standard test packet as determined by hourly test runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest arcs. In the general case, the labels on the arcs could be computed as a function of the distance, energy, mean transmission count, measured delay, and other factors. By changing the weighting function, the shortest path algorithm would then compute the "shortest" path measured according to any one of a number of criteria or to a combination of criteria. In the following, we discuss in brief some of these commonly used routing metrics in WSNs.

1. Minimum Hop Count: The most common metric used in routing protocols is the minimum hop (or shortest hop) where the routing protocol attempts to find the path from the sender to the destination that requires the smallest number of relay nodes (hops). In this simple

technique, every link has the same cost and a minimum-hop routing protocol selects the path that minimizes the total cost of data propagation from source to destination. The basic idea behind this metric is that using the shortest path will result in low end-to-end delays and low resource consumptions, because the smallest possible number of forwarding nodes will be involved. However, since the minimum-hop approach does not consider the actual resource availability on each node, the resulting route is probably non optimal in terms of delay, energy, and congestion avoidance. Nevertheless, the minimum-hop metric is being used in many routing protocols due to its simplicity and its isotonicity, that is, its ability to ensure that the order of weights of two paths is preserved even when they are appended or prefixed by a common third path.

2. Energy: The most critical resource in WSNs is undoubtedly the available energy of sensor nodes. However, there is not one unique energy metric that can be applied to the routing problem; instead, there are various different interpretations of energy efficiency, including (Singh et al. 1998):

   (a) Minimum energy consumed per packet: This is the most natural concept of energy efficiency, that is, the goal is to minimize the total amount of energy expended for the propagation of a single packet from the source node to the destination. The total energy is then the sum of the energy consumed by each node along a route for receiving and transmitting the packet.

   (b) Maximum time to network partition: A network partitions into several smaller subnetworks when the last node that links two parts of the network expires or fails. As a consequence, a subnetwork may not be reachable, rendering the sensor nodes within the subnetwork useless. Therefore, the challenge is to reduce the energy consumption on nodes that are crucial to maintaining a network where every node can be reached via at least one route. For example, a minimal set of nodes, whose removal will cause a network to partition, can be found using the max-flow/min-cut theorem. Once a routing protocol has identified these critical nodes, it can attempt to balance the traffic load such that premature expiration of these nodes is prevented.

   (c) Minimum variance in node power levels: In this scenario, all nodes within the network are considered equally important and the challenge is to distribute the energy consumption across all nodes in the network as equally as possible. The goal of such an approach could be to maximize the lifetime of the entire network, for example, instead of some nodes expiring sooner than others and

thereby continuously decreasing the network size, one could aim at keeping as many nodes alive as long as possible. In the ideal (but practically impossible) case, all nodes would expire at exactly the same time.

(d) Maximum (average) energy capacity: In this approach, the focus is less on the energy cost of packet propagation, but instead on the energy capacity (i.e., the current battery charge level) of the nodes. A routing protocol that uses this metric would then favor routes that have the largest total energy capacity from source to destination. A routing protocol that uses this metric must be carefully designed to avoid the pitfall of choosing unnecessarily long routes in order to maximize the total energy capacity. A variation of this metric is to maximize the average energy capacity, which can avoid this problem.

(e) Maximum minimum energy capacity: Finally, instead of maximizing the energy capacities of the entire path, the primary routing goal could be to select the path with the largest minimum energy capacity. This technique also favors routes with larger energy reserves, but also protects low-capacity nodes from premature expiration.

3. Expected Transmission Count (ETX): ETX, proposed by de Couto et al (2003), is defined as the expected number of MAC layer transmissions necessary to successfully deliver a packet through a wireless link. The weight of a path is defined as the summation of the ETX of all links along the path. ETX is a routing metric that ensures easy calculation of minimum weight paths and loop free routing under all routing protocols. However, it never considers the energy consumption of devices.

4. Expected Transmission Time (ETT): Expected transmission time is an estimation of the time cost of sending a packet successfully through the MAC layer. It takes the link bandwidth into account and it is commonly used to express latency.

5.5 RPL Routing Protocol

This section provides a brief description of the ROLL RPL routing protocol for WSNs. The IETF Routing over Low power and Lossy network (ROLL) working group has recently developed this IPv6 based routing protocol for Low power and Lossy Networks (LLNs). RPL is a Distance vector routing protocol designed for low power and lossy networks and is intended for

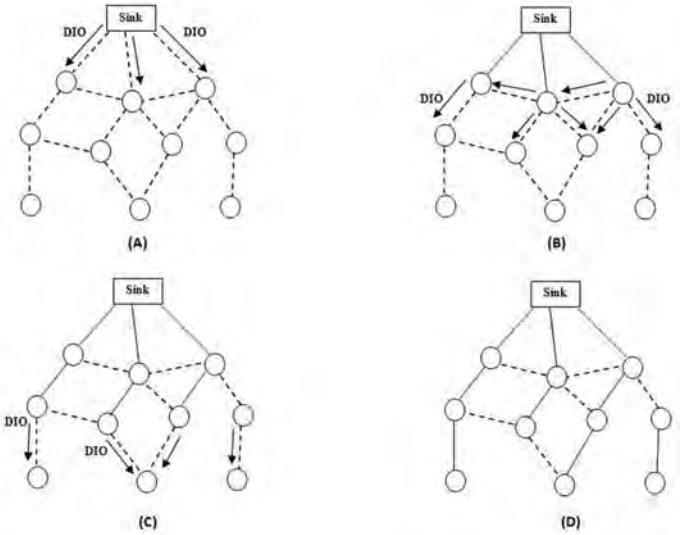


Figure 5.8 The routing protocol RPL DAG topology formation through DIO messages.

- Industrial and home automation;
- Healthcare;
- Smart grids.

RPL allows routing across multiple types of link layers. It constructs a Destination-Oriented Directed Acyclic Graphs (DODAGs), i.e., trees sources-destinations. The nodes build and maintain DODAGs by periodically multicasting messages, the so called DODAG Information Object control message (DIO), to their neighbors. In order to join a DODAG, a node listens to the DIO messages sent by its neighbors and selects a subset of these nodes as its parents. In particular, nodes that receive the messages will process these messages and make decision whether to join the graph or not and also forward the DIO messages to its neighboring nodes. Destination Advertisement Option control messages (DAO) are sent periodically to notify a parent about the routes to its children nodes. This process continuous such that the DAG topology is built from the sink node toward edges nodes as shown in Figure 5.8.

A DIO control message includes the node's rank (its level) d_j and a packet forwarding metric a_{ij} and is broadcasted to build the tree. The metrics used to characterize the cost of each arc in the graph include the link reliability, the packet delay, the node energy consumption, ETX etc. DODAG minimizes the cost to go to the root (destination node) based on an Objective Function. The Objective Function defines how nodes in RPL select the routes within

an instance. It combines the metrics and constraints to find the best path. For example, the objective function finds the path that has minimum delay and that path never traverse a battery-operated node. In this example, the path with minimum delay represents the metric and non-battery operated nodes represent the constraint.

Problems

Problem 5.1 Shortest path routing: Bellman-Ford algorithm (Ex.8.4 in (Pottie and Kaiser, 2005))

The network topology of Figure 5.9 is used to illustrate the Bellman-Ford algorithm for finding the shortest route to a node. In the figure, the number beside a node serves as the label for the node, and the number near an arc indicates the length of the arc. For instance, the arc connecting nodes 1 and 2 has length 1. Define d_{ij} to be the length of the direct arc connecting nodes i and j . If there is no direct arc connecting the two nodes, we set $d_{ij} = \infty$. By doing this, d_{ij} has meaning for any pair of nodes i and j in the network.

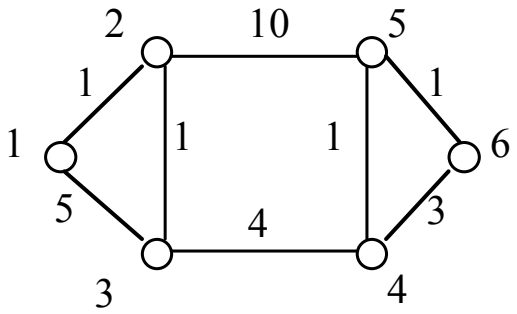


Figure 5.9 A simple network where finding the shortest path.

Consider node 1 as the destination node. The shortest path from node i to node 1 that traverses at most h arcs and goes through node 1 only once is called a shortest ($\leq h$) walk, and its length is denoted by D_i^h. Note there are two special cases. If all paths between node i and 1 consist of more then h arcs, $D_i^h = \infty$. By convention, $D_1^h = 0$ for any h.

(a) Determine D_i^0 for $i = 1, 2, \ldots, 6$. Find d_{ij} for all possible i, $j = 1, 2, \ldots, 6$.

(b) The following iteration is used to generate the subsequent shortest walks:

$$D_i^{h+1} = \min_j [d_{ij} + D_j^h] \quad \text{for all } i \neq 1 .$$

Determine D_i^1 for $i \neq 1$.

(c) Use the iteration equation in (b) to compute D_i^2 , D_i^3 ,... for $i \neq 1$. Stop the iteration when $D_i^{h+1} = D_i^h$, for all $i \neq 1$. The minimum distance from node i to node 1 is D_i^h in the last iteration.

Problem 5.2 Shortest path routing: Dijkstra algorithm (Ex.8.5 in (Pottie and Kaiser, 2005))

We will use Figure 5.9 to illustrate the Dijkstra algorithm for finding the shortest route to a destination node. The length of the direct arc connecting nodes i and j is defined to be d_{ij}. For a detailed description of the figure and the definition of d_{ij}, refer to previous exercise. Denote by P the set of nodes whose shortest path to the destination node is known, and denote by D_j the current shortest distance from node j to the destination node. Note that only when node j belongs to the set P can we say D_j is the true shortest distance. Choose node 1 as the destination node. Initially, set $P = \{1\}$, $D_1 = 0$, and $D_j = \infty$ for $j \neq 1$.

(a) Update D_j for $j \neq 1$ using the following equation

$$D_j = \min[D_j, d_{j1}] .$$

(b) Find i such that

$$D_i = \min_{j \notin P}[D_j] ,$$

and update $P := P \cup \{i\}$.

(c) Update D_j for $j \notin P$ by the following equation

$$D_j := \min[D_j, D_i + d_{ji}] ,$$

in which i is the i obtained in (b).

(d) Go back and compute steps (b) and (c) recursively until P contains all the nodes in the network. The resulting D_j is the shortest distance from node j to node 1.

Problem 5.3 Shortest path routing in WSNs

One way of building routing tree in WSNs is based on ETX. ETX stands for expected number of transmissions. The Idea is to make a minimum spanning tree (MST) minimizing the expected number of transmissions for each node. This is done based on MAC layer functionalities (e.g., PRR). With PRR for each link between (i, j) nodes have a good estimate of packet reception rate from other party and hence can measure the temporal reliability of the link. Note that PRR is directional and the rate of packet reception for links (i, j) and (j, i) can be different. Having the values of PRR of direct neighbors available at each node, in a recursive fashion nodes can build a routing tree that minimizes the expected number of transmissions to the sink.

(a) Develop a sketch of the algorithm and the required equations to build the routing tree based on ETX metric.

(b) Consider Figure 5.10 and assume the PRR is bidirectional (links are undirected) where the values of the PRR are given on the arcs. Find the MST based on ETX metric.

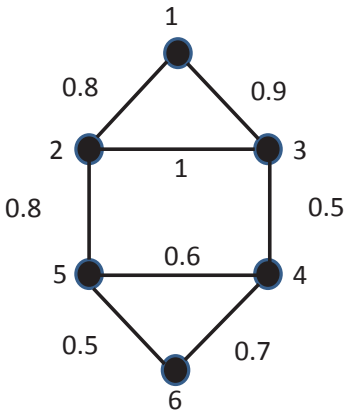


Figure 5.10 A sample WSN topology. Node 1 is the sink and link qualities (PRR) are depicted on each arc.

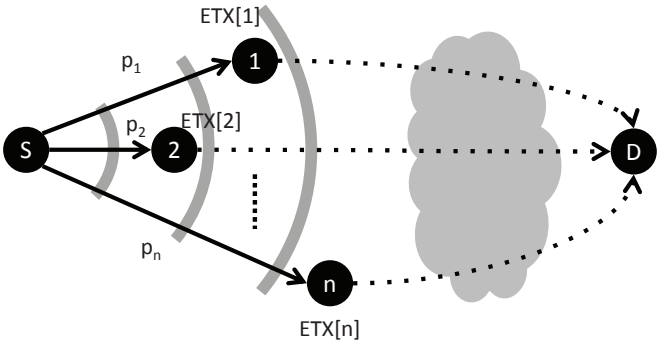


Figure 5.11 *Initial graph with link probabilities annotated. Each neighbor i of node s provides its ETX[i] to s.*

Problem 5.4 Anycast routing over WSNs

In WSNs, the expected number of transmissions of a node (ETX) is a routing metric, namely a metric used by a node to take the decision over which path the node routes packets. Denote by ETX[s] the expected number of transmissions required for node s to send a packet to the destination D. Let $\mathcal{N}_s$, $\mathcal{P}_s$ and p_i be the neighbors set of s, parent set of s and probability of successful transmission from node s to neighboring node i, respectively. Given ETX[i] and p_i for all $i \in \mathcal{N}_s$, ETX at s is defined as

$$\text{ETX}[s] = \min_{i \in \mathcal{N}_s} \left\{ \text{ETX}[i] + \frac{1}{p_i}, \right\}$$

and the parent set of s is defined as $\mathcal{P}_s = \{i\}$,where i is the neighbor that minimizes ETX[s] above. Note that the $\mathcal{P}_s$ has one component.

Now we want to extend this scheme to consider multiple parents. Figure 5.11 illustrates such network. The routing scenario is as follows. Node s looks at its

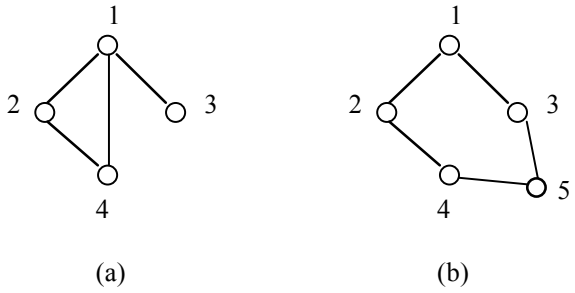


Figure 5.12 Spanning tree.

parents set $\mathcal{P}_s = \{1 \dots n\}$ as an ordered set. It broadcasts a packet to all the parents and waits for an acknowledgement (ack) packet. If parent 1 receives the packet (with probability p_1) then node 1 will forward the packet to D (with cost ETX[1]). Now if node 1 fails to receive the packet and node 2 receives it, then node 2 will forward it. So within this scheme node i is allowed to forward a packet if 1) it successfully receives the packet from s with probability p_i and 2) if all the nodes with higher priority $1, \dots, i-1$ fail to get the packet. Assume that an efficient message passing scheme handles this structure.

(a) Calculate the new ETX metric for s and a given ordered set of parents $\mathcal{P}_s = \{1 \dots n\}$. [hint: first you can calculate the probability that a packet from s is received by at least one of the parents. Then, conditioned on that you are in one of the parents (the first hop transmission is successful), calculate the average ETX from one of the parents to the destination.]

(b) In Figure 5.11, assume that s has 3 neighbors with success probabilities $(p_1, p_2, p_3) = (1/2, 1/3, 1)$ and ETX of $(2, 2, 4)$, respectively. Calculate the ETX$[s]$ for two cases: with single parent and three parents with priority order $(1, 2, 3)$.

(c) For the second case of the previous point, find the optimal parent set (note that there are $2^3 - 1$ possible parent sets) that minimizes ETX$[s]$.

Problem 5.5 Spanning tree (Ex.8.7 in (Pottie and Kaiser, 2005))

Find all possible spanning trees for the two graphs in Figure 5.12 subject to the constraint that node 1 must be the root. Determine the number of nodes N and arcs A in each of these spanning trees. Can you see a relation between N and A?

Problem 5.6 Directed diffusion (Ex.8.8 in (Pottie and Kaiser, 2005))

Consider the situation in Figure 5.13. The solid lines represent transmission links between nodes, and dashed lines indicate boundaries of tiers. Here node A wants to transmit to node D. Suppose the transmission takes the branches within the same tier with one third of the probability of branches in the next tier, and the packets do not back-track. Determine the likelihood of packets flowing through node B and C to reach D.

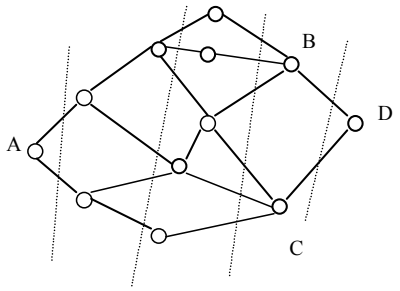


Figure 5.13 Directed diffusion.

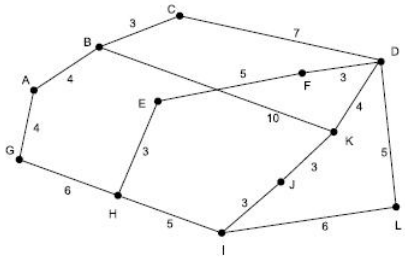


Figure 5.14 The graph for Exercise 5.7

Problem 5.7 Multicast Spanning Tree

Find a possible multicast spanning tree for the graph in Figure 5.14 subject to the constraint that node C must be the root.

Problem 5.8 Bellman-Ford Algorithm

In Bellman-Ford algorithm, the initial values for the paths between the network nodes and the destination node are set equal to ∞. For the network topology in Figure 5.15, show that the algorithm converges for initial path values equal to 0. Consider H as the destination node.

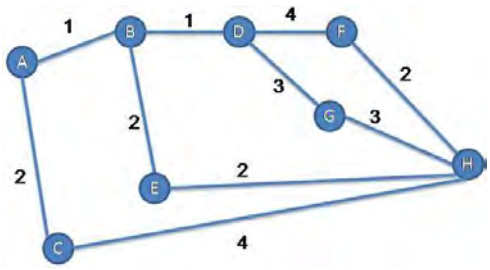


Figure 5.15 Network topology for Exercise 5.8

Chapter 6

Topology Control

6.1 Introduction

Topology control is one of the relevant techniques to decrease the energy consumption and increase the network capacity in WSNs. In this chapter we will introduce the essential concepts of this theory. By taking as reference (Li et al., 2013), we have chosen to split the topology control issues into two categories: *Connectivity* and *Coverage*.

We will first describe a strict model for a WSN and then, in Section 6.1, we formally define what we mean by topology control. In Section 6.2 we consider the issue of connectivity and in Section 6.3 we will consider the coverage issue.

This chapter will contain lots of graph theoretical terms and if not already familiar with these, we encourage the reader to look in the chapter on graph theory in the appendix.

Model

A WSN is modeled as a directed graph $G = (\mathscr{V}, \mathscr{A})$ where each vertex $v \in \mathscr{V}$ represents a node and for any two nodes $u, v \in \mathscr{V}$, $(u, v) \in \mathscr{A}$ if and only if the transmitting range of node u is larger than the distance between u and v. To do this in a formal way, one may think of a WSN as a pair $(\mathscr{N}, p)$ where $\mathscr{N}$ is a set of nodes and $p : \mathscr{N} \to \mathbb{R}^d$ is a *position function* which gives the physical positions of the nodes. Here d refers to the dimension in which me model the WSN, i.e., $d \in \{1, 2, 3\}$. Throughout this chapter we will let δ denote the *euclidean distance function* and if $\mathscr{V}$ is a set then $|\mathscr{E}|$ always denotes the number of elements in $\mathscr{E}$. Furthermore, for the whole chapter we make the following assumption:

Assumption 6.1.1. For any two distinct nodes $n_1, n_2 \in \mathscr{N}$, $p(n_1) \neq p(n_2)$.

Definition 6.1.2. Given a WSN $N_p = (\mathscr{N}, p)$, a *range assignment* for N_p is a function $r : \mathscr{N} \to (0, \rho_{\max}] \subset \mathbb{R}$, assigning to every node v a *trans-*

mitting range $r(v) \in (0, \rho_{\max}]$, where $\rho_{\max}$ is the maximum range of the nodes' transceivers. If $r(v)$ is the same for all $v \in \mathscr{V}$ then we say that r is *homogeneous*.

We also call r a *range assignment for* $\mathscr{N}$ or a *range assignment for* $p(\mathscr{N})$ and may even speak of a range assignment for a subset of $\mathbb{R}^n$ without any underlying WSN.

Definition 6.1.3. Let $N_p = (\mathscr{N}, p)$ be a WSN and $\mathscr{V} = p(\mathscr{N}) \subset \mathbb{R}^n$. Given a range assignment r for $\mathscr{N}$, the *communication graph of* N_p*, induced by* r is the directed graph $\overrightarrow{G_r} = (\mathscr{V}, \mathscr{A}_r)$, where, for every $u, v \in \mathscr{V}$, $(u, v) \in \mathscr{A}_r$ if and only if $r(u) \geq \delta(u, v)$. We denote by $G_r = (\mathscr{V}, \mathscr{E}_r)$ the graph such that $\{u, v\} \in \mathscr{E}_r$ if and only if $(u, v) \in \mathscr{A}_r$ and $(v, u) \in \mathscr{A}_r$. $G_r = (\mathscr{V}, \mathscr{E}_r)$ is called the *undirected communication graph of* N_p*, induced by* r.

When we speak of the *network topology* of the WSN, we simply refer to the arcs/edges in the communication graph. Later in this chapter, the word topology will also include the edges in the so called *coverage graph*.

In WSNs, the nodes will communicate through wireless transceivers. Assume that a node u is transmitting a radio signal that is received by a node v at distance $\delta(u, v) = d$. Denote by P_t the power used by u to transmit the signal and let $P_r(d)$ be the power of the signal when reaching v.

A model widely used for WSN is called *the log-distance path model* which can be described by the following relation.

$$P_r(d) \propto \frac{P_t}{d^\alpha}\,.$$

Here, α is called the *distance-power gradient*. In free space we have that $\alpha = 2$, however in real applications α varies depending on the environment. In urban areas, α is often around 2.7-3.5. Indoors we may even have values of α less than 2, due to reflections and if the conditions are bad α may be as high as 6 (Santi, 2005).

Definition

There is no universal definition of topology control and in different literature, we may find many different versions. Topology control is often referred to not only as the construction but also the *maintenance* of the network topology. However, in this chapter we will mostly discuss the theoretical problems arising from the construction of the topology.

We define topology control informally as follows:

Topology Control is the procedure of controlling the network topology characteristics by deciding the nodes parameters in order to optimize some

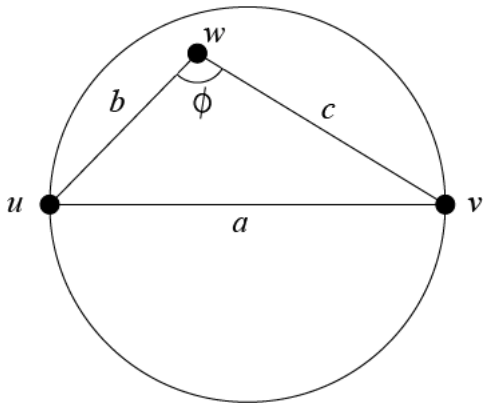


Figure 6.1 When $a^2 \geq b^2 + c^2$, it is more energy-efficient for node u to send via node w to node v, then to send directly to node v.

network performance indicators. Examples of network topology characteristics are the connectivity and coverage. Examples of node parameters are the transmitting range and sleep/awake mode. Examples of network performance indicators are network energy consumption and network capacity.

Motivation

In this section we will give two examples that motivates the use of topology control from different perspectives.

Energy-Efficiency

Suppose that a node u wants to send a message to a node v, and for simplicity, consider the nodes as points in $\mathbb{R}^2$. Suppose that a third node w is located in the disc with diameter $\overline{uv}$. Let $a = \delta(u, v)$, $b = \delta(u, w)$, $c = \delta(v, w)$ and let ϕ be the angle at w in the triangle uvw (see Figure 6.1). From basic geometry we know that $a^2 = b^2 + c^2 - 2bc\cos(\phi)$ and since w is located in the disc with diameter $\overline{uv}$ we have that $\phi \geq \pi/2$. This implies that $\cos(\phi) \leq 0$ and hence $a^2 \geq b^2 + c^2$. If we assume that the distance-power gradient $\alpha = 2$ we conclude that, from an energy-efficiency point of view, it will be better to communicate via w instead of directly from u to v.

Network Capacity

Suppose that 4 nodes, u_1, u_2, u_3, u_4, are located in $\mathbb{R}^3$ and that $\delta(u_1, u_2) \leq \delta(u_2, u_3)$ and $\delta(u_3, u_4) \leq \delta(u_2, u_3)$. Suppose that we are given a range assignment r such that $r(u_2) \geq \delta(u_2, u_3)$ and that u_3 has a message for u_4. As u_2 is transmitting to u_1, u_3 can not transmit the data to u_4 (see Figure 6.2).

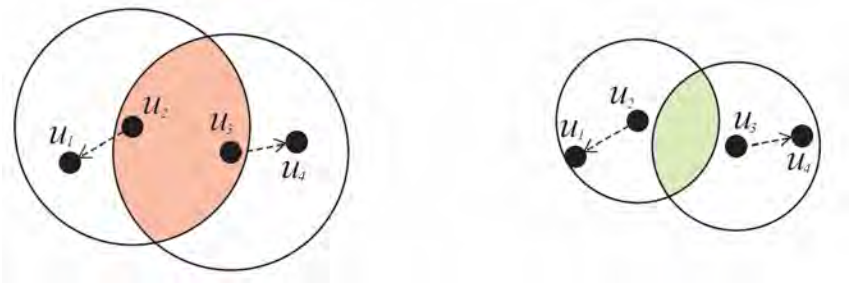


Figure 6.2 The left figure pictures the effect of interference and the right figure pictures a situation without interference.

Suppose one the other hand that $\delta(u_1, u_2) \leq r(u_2) \leq \delta(u_2, u_3)$ and $\delta(u_3, u_4) \leq r(u_3) \leq \delta(u_2, u_3)$. Then both u_2 and u_3 can transmit messages at the same time without causing any interference.

Application

In this section we will discuss two practical examples showing the importance of topology control. These examples relate to the environmental applications of WSN discussed in Section 1.3.

Monitoring CO_2 Emission in Urban Areas

One of the main causes of global warming is the over-emission of CO_2. According *United Nations Framework Convention on Climate Change*, many countries will have to reduce or limit their total emission of CO_2 to stop or at least, slow down the global warming. This has led to an increase in demand for accurate, real-time measuring of the CO_2 emission in urban areas.

A great challenge in this application is the huge size of the area to be covered, which requires a large scale systems. Typically, a WSN covering the area would consist of more than 1000 nodes and should support many different services, such as link estimation, data collection, data processing, and structure management (e.g. layering and clustering). This requires protocols that can support many network services at the same time. A challenge here is to implement such protocols without loosing the scalability and energy-efficiency of the WSN.

In a project called *CitySee* (Mao et al., 2012), more than 1000 sensors where deployed in an urban area of Wuxi City, China, with the objective of monitoring CO_2 emission. In this project, the focus was on node deployment and all nodes had the same transmitting range.

Forest Monitoring

Forest is a very important resource on Earth and researchers can make use of data such as temperature, humidity and illumination for, e.g., forest research, forest surveillance and fire risk evaluation.

To collect the data one can use a WSN. Since the sensors will be deployed in the forest, the batteries should be able to last for a long time. The challenge in terms of topology control is therefore the trade-off between network performance and network lifetime.

In a project called *GreenOrbs*, more than 1000 sensors of different kind are deployed in a forest to operate for more than a year. A first application of GreenOrbs is canopy closure estimates (Mo et al., 2009). A Canopy closure estimate is defined as an estimation of the percentage of ground area vertically shaded by overhead foliage. The sensors are randomly deployed in the forest to measure the illuminance and then examine whether they are in the light or shade.

Design Guidelines

When a WSN becomes very large it is both hard and expensive to have a centralized topology control mechanism, since this requires global information about the WSN. Hence it necessary to introduce the concept of *distributed topology control*. Instead of controlling the whole topology in one piece, one may use a distributed control algorithm such that smaller parts of the network are controlled separately. In this way, the nodes will only need local information about the WSN, which is often easier to manage than using a centralized approach. This is when the scaling property of the communication graph becomes important.

An important aspect due to energy savings is the quality of the information used by the nodes to build the topology. It might be easier to write a protocol when every node knows the physical positions of its neighbors but this might require GPS devices that are expensive in terms of energy. An alternative might be that the nodes know only the distance or the direction to its neighbors, or maybe just the number of neighbors or their identity. Different quality of information is needed in different scenarios.

If the nodes can rely on lower-quality information the application range becomes larger since the WSN are often implemented in harsh environments. An example of this is when WSNs are used to monitor cole mines, since it is well known that GPS devices perform badly under ground.

We will now point out some design guidelines for topology control.

1. **Connectivity:** The WSN must be connected, that is; have a strongly connected communication graph.

2. **Coverage:** The sensors must cover the area of interest.

3. **Distributed algorithm:** Unlike a centralized algorithm, a distributed algorithm is not in need of global information, and is therefore cheaper. This requires scalability and that the nodes can make decisions locally.

4. **Small node degree:** A small node degree means that the corresponding vertex in the communication graph has a small degree. This is important since it reduces the amount of interference.

5. **Energy-Efficiency:** One of the biggest reasons to use topology control is to save energy. Hence it is important that the topology control mechanism does not require more energy than the energy we gain from using it. This point is related to all other points and often the one which puts a spoke in the wheel.

6. **Low-complexity algorithms:** Since the WSN devices are often very simple, the topology control algorithm must have low computational complexity. The simplicity of the topology control mechanism also improves the energy-efficiency.

7. **Local information:** Every node need some information about its neighborhood, for example its physical position or the position of its neighbors. This might require some extra hardware such as GPS devices etc.

8. **Low-quality communication:** The WSN communicates over wireless channels, and is often implemented in harsh environments. This effects the quality of the signals between the nodes.

6.2 Connectivity Problems

An important characteristic of the network topology is that the corresponding communication graph is strongly connected. In this section, we will treat this problem from a few different angles.

6.2.1 Range Assignment Problems

A famous problem concerning the connectivity issue is the so called *range assignment problem*. This can be described as the problem of finding a range assignment, for a set of nodes in a WSN, which minimizes a certain sum which corresponds to the energy needed to set up the network topology.

This approach implies a *centralized control algorithm* since the controller needs global information about the WSN. In Section 6.1 we briefly discussed the concept of *distributed topology control*, which on the other hand requires only local information about the WSN.

Problem Definition

Following the log-distance path model recalled in Section 6.1, we conclude that, to minimize the power used by a WSN $N_p = (\mathscr{N}, p)$ to construct a good topology, an important criterion is to minimize $\sum_{v \in \mathscr{N}} r^\alpha$.

Definition 6.2.1. Given a WSN $N_p = (\mathscr{N}, p)$ and a range assignment r for N_p, the *cost of* r, $C(r)$ is defined by

$$C(r) = \sum_{v \in \mathscr{N}} r(v)^\alpha,$$

where α is defined as the distance-power gradient.

Definition 6.2.2 (The Range Assignment Problem (RA))**.** Given a constant $\beta \in \mathbb{R}$, let $\mathscr{V}$ be a finite subset of $\mathbb{R}^n$. For any range assignment r for $\mathscr{V}$, let $\overrightarrow{G_r} = (\mathscr{V}, \mathscr{A}_r)$ be the communication graph induced by r. Let $\mathcal{R}$ be the set of all range assignments for $\mathscr{V}$ such that $\overrightarrow{G_r}$ satisfies certain connectivity conditions. The problem of finding a range assignment $r_{\min} \in \mathcal{R}$ such that

$$\sum_{v \in \mathscr{V}} r_{\min}(v)^\beta = \min_{r \in \mathcal{R}} \sum_{v \in \mathscr{V}} r(v)^\beta$$

is called *the range assignment problem* (RA).

We will know define 4 RA-problems with different constraints on the communication graph $\overrightarrow{G_r} = (\mathscr{V}, \mathscr{A}_r)$. Note that for these 4 problems we assume that $\rho_{\max} \to \infty$.

Connectivity Conditions:

1. Connectivity: G_r is connected.

2. Strong connectivity: $\overrightarrow{G_r}$ is strongly connected.

3. Broadcast: Given a vertex $v \in \mathscr{V}$ (before choosing the range assignment), $\overrightarrow{G_r}$ contains an arborescence rooted at v.

4. Symmetry: $\overrightarrow{G_r}$ is bidirectional and strongly connected.

Note that problems 1, 2, 3 are of decreasing strength, i.e. if G_r is connected then $\overrightarrow{G_r}$ is strongly connected, and if $\overrightarrow{G_r}$ is strongly connected then, for any $v \in \mathscr{V}$, $\overrightarrow{G_r}$ contains an arborescence rooted at v. Problem 4 is of a

different nature and implies 1, 2 and 3. An equivalent formulation of 4 is $\overrightarrow{G_r}$ is bidirectional and G_r is connected.

We are going to show that, with certain constraints on n and β, problems 1, 2, 3, 4 are *NP-hard*. For an introduction to complexity theory we recommend Garey and Johnson's classical: "*A Guide to the Theory of NP-Completeness* " (Garey and Johnson, 1979).

A very quick and informal summation what NP-hard means can be given as follows:

- A *decision problem* is on the form: INSTANCE: {...}, QUESTION: {...} and the solution is either "YES" or "NO".
- NP - Class of decision problems, whose "yes"-instances could be verified in polynomial time by a non-deterministic Turing machine.
- NP-hard - "At least as hard as the hardest problem in NP".

By the last by we mean that a a solution to problem that is NP-hard can be used to solve one of the "hardest" problems in NP.

What we typically do when we want to show that a problem A is NP-hard is to take a problem B which is known to be NP-hard and make a construction realizable in polynomial time such that solving problem A in polynomial time would imply that we could also solve problem B in polynomial time. This would imply that problem A is, so to speak, "at least as hard as problem B" and hence problem A is NP-hard.

To be completely correct here, we would have to formulate the RA-problem as a decision problem. However, we will allow ourselves to be a bit informal on this point. It is easy to see that it would be at least as hard to solve the RA-problem, as to solve the following RA-decision problem:

The Range Assignment Decision Problem (RAD):

Instance: Constants $\beta, n, B \in \mathbb{R}$ and a finite set $\mathscr{V} \subset \mathbb{R}^n$.

Question: Is there a range assignment r for $\mathscr{V}$ such that the communication graph induced by r satisfies a certain *connectivity condition* and

$$\sum_{v \in \mathscr{V}} r_{\min}(v)^\beta \leq B$$

A construction we will use in some proofs is what is called an *orthogonal grid drawing*.

Algorithm 1: Assigns a distance to each vertex in the tree rooted at u.

Result: When making the call Search$(u, y = 0)$ is called, every vertex v is assigned a distance $dist(v)$, to the root u.

Function `Search`(u, y)**:**

```
dist(u) = y;
y = y + 1;
if {u, right(u)} ∈ E_T then
    Search(right(u), y);
if {u, mid(u)} ∈ E_T then
    Search(mid(u), y);
if {u, left(u)} ∈ E_T then
    Search(left(u), y);
```

Definition 6.2.3. Let G be a planar graph of maximum degree 4. An *orthogonal grid drawing of* G is a plane graph $D(G)$ isomorphic to G, and such that the vertices of $D(G)$ have integer coordinates and edges of $D(G)$ are orthogonal gridlines.

Lemma 6.2.4. *Let G be a planar graph of maximum degree 4. There exists an orthogonal grid drawing of G, computable in polynomial time.*

Sketch of proof. Suppose that $G' = (\mathscr{V}, \mathscr{E})$ is any plane graph isomorphic to G and let $\mathscr{F}(G')$ be the set of faces of G'. Pick a vertex $u \in \mathscr{V}$ and construct a spanning tree $T = (\mathscr{V}, \mathscr{E}_T)$ rooted at u. Let $f_c \in \mathscr{F}(G')$ be the outer face of G'.

For any face $f \in \mathscr{F}(G')$ we denote by $J(f)$, (jumps) the number of edges in $\mathscr{E} \setminus \mathscr{E}_T$ that we need to cross to get from f to f_c.

We will add an *orientation* for the edges of G'. Let w be any node in $\mathscr{V}$ and let e_1 be the edge at w, which is contained in the unique path in T from u to w. If $e = \{w', w\}$, and we move clockwise around w starting at e, will pass maximum 3 neighboring vertices and we will call the first one *right*(w), the second one *mid*(w) and the third one *left*(w). If for example, there is no second one, *mid*(w) and *left(w)* does not exist.

For every vertex v in T we assign a *distance* (number of edges) from v to the root u by Algorithm 1.

We also add a counter-clockwise order to the leafs in T.

Figure 6.3 illustrates how an orthogonal drawing of a tree is constructed as follows:

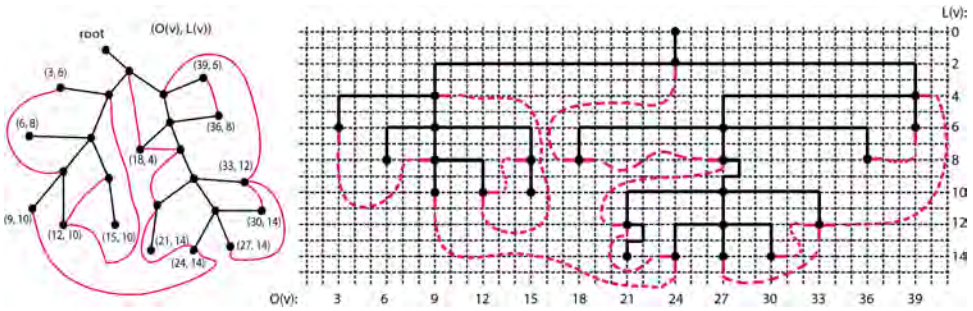


Figure 6.3 Spanning tree (left) and orthogonal drawing of the spanning tree (right).

1. For each leaf v let $O(v) = 3 \times \mathit{order}(v)$ and for each vertex v' let $L(v') = 2 \times \mathit{dist}(v')$. ($O$ = order, L = level)

2. Take a grid where the number of vertical lines are greater than $\max_{\mathit{leaf}\ v} O(v)$, and the number of horizontal lines equals $1 + \max_{v \in \mathscr{V}} L(v)$.

3. Number the vertical lines by the order of the leafs, and the horizontal lines by levels as in Figure 6.3. This gives us a coordinate system where each leaf v' has a coordinate $(O(v'), L(v'))$.

4. Add each leaf to its coordinate in the grid. Note that, as soon as a non-leaf vertex w is added to the grid it is assigned an order $O(w)$.

5. Starting from the highest level, repeat the following for every level L in the grid:

   i) Suppose that three vertices v_1, v_2, v_3 on the current level L, with $O(v_1) < O(v_2) < O(v_3)$, have a common neighbor on level $L-2$. Then add this neighbor at $(O(v_2), L-2)$ and add a vertical gridline for each v_i starting at $(O(v_i), L(v_i))$ and ending in $(O(v_i), L(v_i)-2)$, and add a horizontal gridline starting at $(O(v_1), L-2)$ and ending in $(O(v_3), L-2)$.

   ii) Suppose that two vertices w_1, w_2 have a common neighbor w on level $L-2$ and $O(w_1) < O(w_2)$. Add vertical gridlines from $(O(w_i), L)$ to $(O(w_i), L-2)$ and add a horizontal gridline from $(O(w_1), L-2)$ to $(O(w_2), L-2)$.

   If $\deg(w) = 4$ in G' do the following: If $\{w, \mathit{right}(w)\} \in \mathscr{E} \setminus \mathscr{E}_T$ then add w at $(O(w_2), L-2)$. If $\{w, \mathit{mid}(w)\} \in \mathscr{E} \setminus \mathscr{E}_T$ then add w at $(\lceil (O(w_2) + O(w_1))/2 \rceil, L-2)$. If $\{w, \mathit{left}(w)\} \in \mathscr{E} \setminus \mathscr{E}_T$ then add w at $(O(w_1), L-2)$.

   If $\deg(w) = 3$ in G', then add w at $(O(w_1), L-2)$.

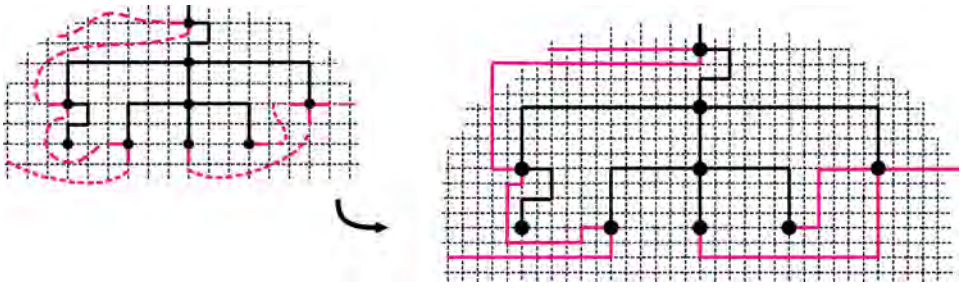


Figure 6.4 Part of an orthogonal drawing of a spanning tree to the left and to the right we see the same part scaled by a factor 2 and the edges of the original graph are mapped to gridlines.

iii) Suppose that a vertex w has a neighbor w' on level $L-2$ and that w' is not a neighbor in T of any other vertex than w. Then add w' at $(O(w), L-2)$.

If $\deg(w') \leq 3$ in G', then add a vertical gridline from $(O(w), L)$ to $(O(w), L-2)$

If $\deg(w') = 4$ in G' and $\{w', \mathit{mid}(w')\} \in \mathscr{E}_T$, then add a vertical gridline from $(O(w), L)$ to $(O(w), L-2)$.

If $\deg(w') = 4$ in G' and $\{w', \mathit{left}(w')\} \in \mathscr{E}_T$, then add a vertical gridline from $(O(w)-1, L)$ to $(O(w)-1, L-2)$, a horizontal gridline from $(O(w), L)$ to $(O(w)-1, L)$ and a horizontal gridline from $(O(w)-1, L-2)$ to $(O(w), L-2)$.

If $\deg(w') = 4$ in G' and $\{w', \mathit{left}(w')\} \in \mathscr{E}_T$, then add a vertical gridline from $(O(w)+1, L)$ to $(O(w)+1, L-2)$, a horizontal gridline from $(O(w), L)$ to $(O(w)-1, L)$ and a horizontal gridline from $(O(w)+1, L-2)$ to $(O(w), L-2)$.

6. The gridlines added in step 5 are the edges of $D(T)$. No two edges in $D(T)$ intersect.

Now we will add the edges in $\mathscr{E} \setminus \mathscr{E}_T$ to the orthogonal drawing as shows in Figure 6.4. We do this is the following order preserving the orientation of the edges: First we will add every edge $e \in \mathscr{E} \setminus \mathscr{E}_T$ bounding a face f with

$$J(f) = \max_{f' \in \mathscr{F}(G')} J(f') \,.$$

Then we add every edge bounding a face f' with $J(f') = J(f) - 1$ and so on, until all edges in $\mathscr{E} \setminus \mathscr{E}_T$ are added.

In this way, if we do not care about the grid for a while, we can draw the edges in $\mathscr{E} \setminus \mathscr{E}_T$ such that no edges cross. By construction, there is always a "way out" in the grid from each node where an edge in $\mathscr{E} \setminus \mathscr{E}_T$ is to be

added. Now it is not hard to see that, by scaling the grid drawing by a proper factor, we may adjust the lines corresponding to edges in $\mathscr{E} \setminus \mathscr{E}_T$ such that they are also gridlines and we may do this without any edges crossing (see Figure 6.4).

□

There are algorithms which are far more sophisticated than the one we just showed. The purpose of this quite sketchy proof is just to give an intuitive feeling that every planar graph of maximum degree 4 has an orthogonal grid drawing. For a strict proof of a stronger result see (Tamassia and Tollis, 1989).

We are ready for a first result of the RA-problem. The proof of the following theorem follows the proof in (Fuchs, 2006).

Theorem 6.2.5. *Solving* **Connectivity** *for $n = 2$ and $\beta > 0$ is NP-hard.*

Proof. The idea of this proof is to take the problem of finding a minimal vertex cover for a graph with maximum degree 3 (which is known to be NP-complete) and make a construction realizable in polynomial time such that solving the RA problem in polynomial time would imply that we could also solve this graph problem in polynomial time. This would imply that the RA problem is NP-hard.

Let $G_3 = (\mathscr{V}, \mathscr{E})$ be any planar graph of maximum degree 3. We will use a construction which is similar to a construction that Gary and Johnson used in (Garey and Johnson, 1977) in the reduction from 3-planar vertex cover to 4-planar connected vertex cover.

1. For each $e \in \mathscr{E}$ add two vertices a_e, b_e (to the interior of e) to split e into three parts. Let $\mathscr{V}_1 = \{a_e, b_e : e \in \mathscr{E}\}$, $\mathscr{E}_1 = \big\{\{u, a_e\}, \{a_e, b_e\}, \{b_e, v\} : e = \{u, v\} \in \mathscr{E}\big\}$ and let $G_3' = (\mathscr{V} \cup \mathscr{V}_1, \mathscr{E}_1)$.

2. For each $u \in \mathscr{V}_1$, let $\mathscr{F}_u$ be the set of faces adjacent to u. Hence $|\mathscr{F}_u| \in \{1, 2\}$. For each $f \in \mathscr{F}_u$ add one vertex $v_{u,f}$, let $\mathscr{V}_2 = \{v_{u,f} : f \in \mathscr{F}_u, u \in \mathscr{V}_1\}$ and $\mathscr{E}_2 = \{\{u, v_{u,f}\} : f \in \mathscr{F}_u, u \in \mathscr{V}_1\}$.

3. For each $x \in \mathscr{V}$ add one vertex $v_{x,f}$ to any face f_x adjacent to x. Let $\mathscr{V}_3 = \{v_{x,f_x} : x \in \mathscr{V}\}$ and $\mathscr{E}_3 = \{\{x, v_{x,f_x}\} : x \in \mathscr{V}\}$.

4. For each face f of G_3 connect all $v_{u,f} \in \mathscr{V}_4 = \mathscr{V}_2 \cup \mathscr{V}_3$ by a single cycle, by adding an edge-set $\mathscr{E}_f$, in such a way that the graph remains planar. Let $\mathscr{E}_4 = \bigcup_f \mathscr{E}_f$.

Figure 6.5 illustrates steps 1,2,3,4. Let $G_4 = \{\mathscr{V} \cup \mathscr{V}_1 \cup \mathscr{V}_4, \mathscr{E}_1 \cup \mathscr{E}_2 \cup \mathscr{E}_3 \cup \mathscr{E}_4\}$ be the resulting 4-planar graph. Construct a plane orthogonal drawing $D(G_4)$ of G_4 (by Lemma 6.2.4 this is possible) and scale it by a factor 3. Now we will add vertices to $D(G_4)$ in the following way:

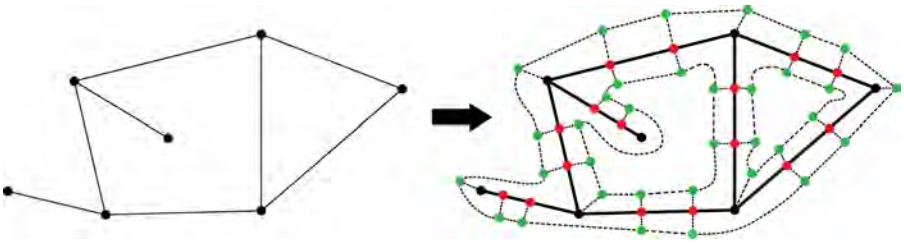


Figure 6.5 Construction of G_4. Red vertices corresponds to step 1, and green vertices corresponds to steps 2 and 3.

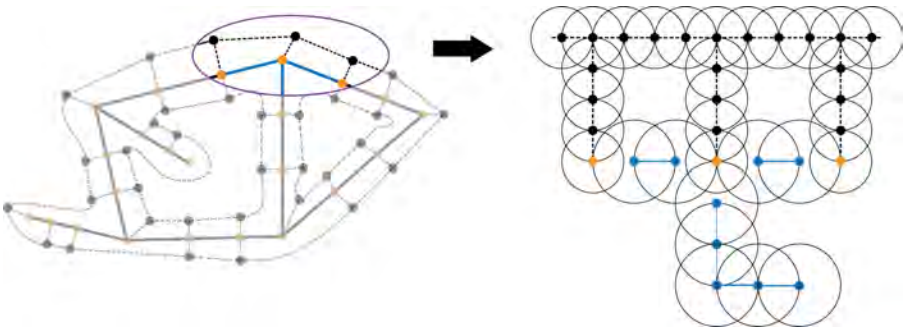


Figure 6.6 Plane orthogonal drawing of G_4 with vertices corresponding to G'_3 in orange and disconnected components corresponding to edges in G'_3 in blue.

a) For each edge corresponding to an edge in $\mathscr{E}_1$, place vertices at the end-points and at points with distance 1 from each other.

b) For each edge corresponding to an edge in $\mathscr{E}_2 \cup \mathscr{E}_3 \cup \mathscr{E}_4$, place vertices at the end-points and at points with distance 3/4 from each other.

Figure 6.5 illustrates steps a) and b)a. Now let $\mathscr{N}$ be the set of vertices added in steps **a)** and **b)** and let $r_{\min}$ be the minimal range assignment for $\mathscr{N}$ such that each $n \in \mathscr{N}$ is connected to its nearest neighbor, i.e. for any $n \in \mathscr{N}$, $r_{\min}(n) = \min_{n' \in \mathscr{N}} \delta(n, n')$.

Hence, for any vertex w added in step **b)**, $r_{\min}(w) = 3/4$, and for any vertex w' added in step **a)** and such that w' is not incident with any grid-line in $D(G_4)$ corresponding to an edge in $\mathscr{E}_2 \cup \mathscr{E}_3 \cup \mathscr{E}_4$, $r_{\min}(w') = 1$. That is, if w' does not correspond to a vertex in G'_3.

The undirected communication graph $G_{r_{\min}}$, induced on $\mathscr{N}$ by $r_{\min}$, consists of $1 + |\mathscr{E}_1|$ components. $|\mathscr{E}_1|$ of these corresponds to the edges in $\mathscr{E}_1$ and the last one corresponds to the rest of the construction.

It is not hard to see that the cheapest way to connect $G_{r_{\min}}$ is to increase the range for some vertices in $\mathscr{N}$ corresponding to vertices in G'_3. Hence, to connect all the $|\mathscr{E}_1|$ components to the rest of the construction, we must

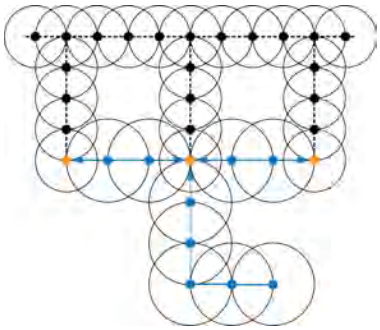


Figure 6.7 Plane orthogonal directed drawing of G_4 with vertices corresponding to G'_3 in orange and disconnected components corresponding to edges in G'_3 in blue.

pick a set of vertices which form a vertex cover for G'_3. So if we can find a minimal range assignment r for $\mathcal{N}$ such that G_r is connected, then we have increased the range for a set of vertices which form a minimal vertex cover of G'_3. Hence, if we can prove the following claim, we are done.

Claim: $G_3 = (\mathcal{V}, \mathcal{E})$ has a vertex cover of size k if and only if $G'_3 = (\mathcal{V} \cup \mathcal{V}_1, \mathcal{E}_1)$ has a vertex cover of size $k + |\mathcal{E}|$.

To prove the claim, first assume that $G_3 = (\mathcal{V}, \mathcal{E})$ has a vertex cover C of size k. Pick any $e = \{u, v\} \in \mathcal{E}$. Then either $u \in C$ or $v \in C$. In $\mathcal{E}_1$, e is divided into three edges $\{u, a_e\}, \{a_e, b_e\}$ and $\{b_e, v\}$. Either $\{u, a_e\}$ or $\{b_e, v\}$ is already covered by C and hence we need to add exactly one vertex (a_e or b_e) to C to cover $\{u, a_e\}, \{a_e, b_e\}$ and $\{b_e, v\}$. This means that we must add one vertex per edge in $\mathcal{E}$ and hence G'_3 has a vertex cover of size $k + |\mathcal{E}|$.

Conversely, suppose G'_3 has a vertex cover C' of size $k + |\mathcal{E}|$. Pick any edge $e = \{u, v\} \in \mathcal{E}$. Then e is split into $\{u, a_e\}, \{a_e, b_e\}$ and $\{b_e, v\}$ in $\mathcal{E}_1$. We always need at least two vertices to cover $\{u, a_e\}, \{a_e, b_e\}$ and $\{b_e, v\}$, but only one vertex to cover e. Hence there is a vertex cover for G_3 of size $k + |\mathcal{E}| - |\mathcal{E}| = k$. □

Theorem 6.2.6. *Solving* **Strong connectivity** *for $n \in \{2, 3\}$ and $\beta > 0$ is NP-hard.*

Proof. It is enough to prove the theorem for $n = 2$. We will use the exact same construction as in Theorem 6.2.5. This time the situation is slightly different since we are dealing with a directed graph (note the arrows in Figure 6.7).

For some values on β it might be more energy-efficient to increase the range for vertices not corresponding to vertices in G'_3. Still, we have to pick one vertex per edge in G'_3 and each vertex will be next to a vertex in G'_3. Hence we get a minimal vertex cover of G'_3. □

Since the *RA* problem is NP-hard, instead of trying to find the optimal solution, people work on finding approximations. One simple approximation is found by constructing a MST on the set of nodes. This can be described as follows.

Definition 6.2.7. Given a WSN $N_p = (\mathscr{N}, p)$, the *MST-approximation* is the range assignment r_T defined as follows:

1. Let $\mathscr{V} = p(\mathscr{N})$ and $G = (\mathscr{V}, \mathscr{E}, w)$ the complete edge-weighted graph where for every $\{u, v\} \in \mathscr{E}$, $w(u, v) = \delta(u, v)^\alpha$.
2. Find a minimal spanning tree $T = (\mathscr{V}, \mathscr{E}_T)$ of G.
3. Let r_T be the range assignment for N_p such that for any $u \in \mathscr{V}$, $r_T(u) = \max_{\{u,v\} \in \mathscr{E}_T} \delta(u, v)$.

Theorem 6.2.8 ((Kirousis et al., 2000))**.** *Let $N_p = (\mathscr{N}, p)$, $\mathscr{V} = p(\mathscr{N}) \subset \mathbb{R}^n$ and let r_T be the MST-approximation. Let r_{opt} be the optimal solution for* **Strong connectivity** *with $n \in \{2, 3\}$ and $\beta = \alpha \geq 1$. Then*

$$C(r_{\text{opt}}) \leq C(r_T) < 2C(r_{\text{opt}}).$$

Proof. First of all, if $|\mathscr{V}| \leq 2$, then $r_T = r_{\text{opt}}$ and the result is obvious, so assume that $|\mathscr{V}| \geq 3$. Let T be the MST arising in the construction of r_T. The proof will be in two steps. First we show that $C(T) < C(r_{\text{opt}})$ and secondly we show that $C(r_T) < 2C(T)$.

First, let $\overrightarrow{G}_{\text{opt}} = (\mathscr{V}, \mathscr{A}_{r_{\text{opt}}})$ be the communication graph induced by r_{opt}. Then $\overrightarrow{G}_{\text{opt}}$ is strongly connected. Let $\overrightarrow{T}'$ be any arborescence of $\overrightarrow{G}_{\text{opt}}$. Since $C(\overrightarrow{T}')$ is a sum over $|\mathscr{V}| - 1$ edges and $C(r_{\text{opt}})$ is a sum over $|\mathscr{V}|$ vertices it is easy to see that $C(\overrightarrow{T}') < C(r_{\text{opt}})$ and of course $C(T) \leq C(\overrightarrow{T}')$.

Secondly, in $C(r_T)$ we will pick one edge for every vertex and each edge can be picked at most twice, since every edge has two end points. This counts to $|\mathscr{E}_T| + 1$ edges. In the sum $2C(T)$ we pick every edge exactly twice, i.e. $2|\mathscr{E}_T|$ edges in total. Formally we can write this as

$$C(r_T) = \sum_{u \in \mathscr{V}} \max_{\{u,v\} \in \mathscr{E}_T} \delta(u, v)^\alpha < \sum_{u \in \mathscr{V}} \sum_{\{u,v\} \in \mathscr{E}_T} \delta(u, v)^\alpha = 2C(T).$$

This concludes the proof. □

Symmetric range assignments

Even though it is technically possible to implement unidirectional links in WSNs this might often not be the best solution. According to (Marina and Das, 2002), marginal benefit of using a high-overhead routing protocol

to utilize unidirectional links is questionable. It seems easier to achieve good performance by simply avoiding unidirectional links.

In **Connectivity** we require that the nodes can communicate only via bidirectional links whereas in **Symmetry**, we allow only bidirectional links. A range assignment of the later kind is called *symmetric*.

Theorem 6.2.9. *Solving* **Symmetry** *for* $n \in \{2, 3\}$ *and* $\beta = 2$ *is NP-hard*

For a proof of Theorem 6.2.9, see for example (Santi, 2005) page 78-85.

6.2.2 Unicast and Broadcast Topologies

Another approach to the problem of finding an energy-efficient topology is to focus on the communication in the network. We will study this in two different ways:

1. **Unicast:** Pick two arbitrary nodes at a time and see how they communicate.

2. **Broadcast:** Pick one arbitrary node at a time and see how it communicates with all other nodes.

Starting out with the topology, obtained when all nodes transmit at maximum power, we will give some tools to analyze how well a reduced topology performs compared to the original topology in the view of unicast versus broadcast.

Definition 6.2.10. Given a WSN, $N_p = (\mathscr{N}, p)$ and a range assignment $r_{\max}$ such that for any $v \in \mathscr{N}$, $r_{\max}(v) = \rho_{\max}$. The *max-power graph*, $\overrightarrow{G}_{\max} = (\mathscr{V}, \mathscr{A}_{\max})$, is the communication graph of N_p induced by $r_{\max}$. The *undirected max-power graph* $G_{\max}$ is the undirected communication graph corresponding to $\overrightarrow{G}_{\max}$.

Unicast and Proximity Graphs

Suppose that we are given a WSN $N_p = (\mathscr{N}, p)$ and the communication graph $\overrightarrow{G_r} = (\mathscr{V}, \mathscr{A}_r)$, with $\mathscr{V} = p(\mathscr{N})$. When a node $n \in \mathscr{N}$ is sending a message to a node $n' \in \mathscr{N}$, this induces a directed path $P = n_0 n_1 \dots n_k$, where $n_0 = n$ and $n_k = n'$.

Suppose now that we are given the max-power graph $\overrightarrow{G}_{\max}$ and a subgraph $\overrightarrow{G}$. A way to measure if $\overrightarrow{G}$ is a good communication graph for N_p is to check how much we loose in energy-efficiency, when sending a message from one node to another, in the worst case.

Definition 6.2.11. Let $\overrightarrow{G}$ be a directed graph and $P = n_0 n_1 \dots n_k n_{k+1}$ a directed path in $\overrightarrow{G}$. The *power cost* of P is defined by

$$C(P) = \sum_{i=0}^{k} \delta(n_i, n_{i+1})^{\alpha},$$

i.e. $C(P)$ is the cost of the graph P with arc-weight $w((u,v)) = \delta(u,v)^{\alpha}$ for any arc (u,v) in P.

Definition 6.2.12. Let $\overrightarrow{G} = (\mathscr{V}, \mathscr{A})$ be a directed graph. Given $u, v \in \mathscr{V}$, a *minimal power path* from u to v, denoted by $P^{G}_{\min}(u,v)$, is a directed path from u to v such that for any directed path P, from u to v, we have that $C(P^{G}_{\min}(u,v)) \leq C(P)$.

Definition 6.2.13. Let $\overrightarrow{G}$ be any subgraph of the max-power graph $\overrightarrow{G}_{\max}$. The *power stretch factor* of $\overrightarrow{G}$ with respect to $\overrightarrow{G}_{\max}$, denoted by $\mathcal{E}(\overrightarrow{G})$, is defined by

$$\mathcal{E}(\overrightarrow{G}) = \max_{u,v \in \mathscr{V}} \frac{C(P^{\overrightarrow{G}}_{\min}(u,v))}{C(P^{\overrightarrow{G}_{\max}}_{\min}(u,v))}.$$

In Section 6.1 we discussed the importance of having a distributed control algorithm. *Proximity graphs* originates from computational geometry and are built only concerning the nearest neighbors of each node (which explains the name *proximity* graphs). Hence, these can be constructed in a fully distributed and localized way. Examples of proximity graphs are the *Relative neighbor Graph* (RNG), the *Gabriel Graph* (GG), the *Delaunay Graph/Triangulation* (DG/DT) and the *Yao Graph with parameter k* (YG_k) (for formal definitions of these graphs, see Section E.2 in the Appendix).

These four graphs have been shown be to be good graphs for WSNs, especially when it comes to low power stretch factor, low average node degree and low maximum node degree.

To get a little familiar with the proximity graphs mentioned above, we will show some relations between then. These are stated in Proposition 6.2.14 and Proposition 6.2.15.

Proposition 6.2.14. *$EMST \subseteq RNG \subseteq GG \subseteq DG$.*

Proof. *$EMST \subseteq RNG$*: Given a set $\mathscr{V}$ of vertices and the complete graph $G_c = (\mathscr{V}, \mathscr{E}_c)$, we want to show that if $T = (\mathscr{V}, \mathscr{E})$ is a minimal spanning tree of G_c, then $\{u,v\} \in \mathscr{E}$ implies that $lune(u,v)$ contains no vertices in its interior.

Suppose to get a contradiction that there is a $c \in \mathscr{V}$ such that $c \in lune(u,v)$. Since T is a tree and $\{u,v\} \in \mathscr{E}$, there is a unique path P, starting at c and ending in either u or v. Assume, without loss of generality,

Algorithm 2: Constructs the Gabriel Graph on a set of nodes.

Data: $\mathscr{N}$ - set of nodes
For any node u, $\mathscr{E}_{\max}(u)$ are the edges in $G_{\max}$ at u and $\mathscr{E}_{GG}(u)$ are the edges in $GG(G_{\max})$ at u.
For any two nodes u, v, $\text{disc}(u, v)$ is the disc with diameter $\overline{uv}$.

Result: For each $u \in \mathscr{N}$ a set $\mathscr{E}_{GG}(u)$ with all edges at u of the Gabriel Graph on $\mathscr{N}$.

```
Initialize;
for u ∈ 𝒩 do
    u broadcast message (ID_u, (x_u, y_u)) at maximum power;
    𝓔_max(u) = 𝓔_GG(u) = ∅;
    if u recieves message (ID_v, (x_v, y_v)) then
        𝓔_max(u) = 𝓔_max(u) ∪ {v};
        if ¬∃w ∈ 𝒩 such that {u, v} ∈ 𝓔_max(u) and w ∈ disc(u, v)
        then
            𝓔_GG(u) = 𝓔_GG(u) ∪ {{u, v}};
        for {u, w'} ∈ 𝓔_GG(u) do
            if v ∈ disc(u, v') then
                𝓔_GG(u) = 𝓔_GG(u) \ {{u, v'}};
```

that P ends in v. Then if we remove $\{u, v\}$ from $\mathscr{E}$ and add $\{v, c\}$, we get a spanning tree of smaller cost. This is a contradiction.

$RNG \subseteq GG$: For any two vertices u, v, the disc with diameter $\overline{uv}$ is contained in $lune(u, v)$.

$GG \subseteq DG$: This is an obvious consequence of part 2 of Theorem E.2.11. □

Proposition 6.2.15. *$RNG \subseteq YG_k$, for $k \geq 6$.*

Proof. Let $\mathscr{V}$ be a set of vertices. Let $YG_k(\mathscr{V}) = (\mathscr{V}, \mathscr{E})$ be the Yao graph of $\mathscr{V}$ and suppose that $\{u, v\} \in \mathscr{E}$. Let $\mathscr{D}$ be the disc of radius $\delta(u, v)$ centered at u. Suppose that $\mathscr{D}$ is divided into k sectors, each with angle $2\pi/k$. Let $\mathscr{S}_v$ be the sector having v on its boundary. It is easy to see that $\mathscr{S}_v \subseteq lune(u, v)$ if $k \geq 6$. □

Algorithm 2 is found in (Song et al., 2004) and constructs the Gabriel graph on a set of nodes in a fully distributed and localized way, assuming that each node knows its physical position.

If $G_{\max} = (\mathscr{V}, \mathscr{E})$, let $GG(G_{\max})$ denote the intersection between $GG(\mathscr{V})$ and $G_{\max}$, and use the same notation for any proximity graph. The results

Graph	Power stretch factor
$RNG(G_{\max})$	$n-1$
$GG(G_{\max})$	1
$DT(G_{\max})$	$\left(\frac{1+\sqrt{5}}{2}\pi\right)^{\alpha}$
$YG_k(G_{\max})$	$\frac{1}{1-(2\sin\frac{\pi}{k})^{\alpha}}$

Table 6.1 Power stretch factor for proximity graphs.

presented in Table 6.1 are showed in (Li et al., 2003). Here we will only prove one of then, stated in Proposition 6.2.16

Proposition 6.2.16. *If $\alpha \geq 2$ then $\mathcal{E}(GG(G_{\max})) = 1$.*

Proof. Let $G_{\max} = (\mathscr{V}, \mathscr{E})$ and let u, v be any two vertices in $\mathscr{V}$. Let P be minimal power path in $G_{\max}$ from u to u' and let $\{v, v'\}$ be any edge in P. Then vv' is a minimal power path from v to v' and hence the disc with $\overline{vv'}$ as diameter can not contain any other vertices. □

Broadcast and Arborescences

When a node n is broadcasting a message in a WSN N_p with a range assignment r, this induces an arborescence $\overrightarrow{T}$, rooted at n. By analyzing $\overrightarrow{T}$ we get a clue how energy-efficient the broadcasting is in $\overrightarrow{G_r}$ compared to in the max-power graph $\overrightarrow{G}_{\max}$ of N_p.

Definition 6.2.17. Given an arborescence $\overrightarrow{T} = (\mathscr{V}, \mathscr{A})$, we define a function $C_b : \mathscr{V} \to \mathbb{R}$ in the following way:

1. If $u \in \mathscr{V}$ is a leaf, then $C_b(u) = 0$,
2. If $v \in \mathscr{V}$ is not a leaf, then $C_b(v) = \max_{(v,v') \in \mathscr{E}} \delta(v, v')^{\alpha}$.

The *broadcast cost* of $\overrightarrow{T}$ is defined by $\sum_{v \in \mathscr{V}} C_b(v)$ and denoted by $C_b(\overrightarrow{T})$.

Suppose that we are given the max-power graph $\overrightarrow{G}_{\max}$ and a subgraph $\overrightarrow{G} \subseteq \overrightarrow{G}_{\max}$. We will now define the *broadcast stretch factor* which measures the energy-efficiency of broadcast communication in $\overrightarrow{G}$ compared to $\overrightarrow{G}_{\max}$ in the worst case.

Definition 6.2.18. Let $\overrightarrow{G}$ be any subgraph of the max-power graph $\overrightarrow{G}_{\max} = (\mathscr{V}, \mathscr{A}_{\max})$. The *broadcast stretch factor* of $\overrightarrow{G}$ with respect to $\overrightarrow{G}_{\max}$, denotes

as $\epsilon(\overrightarrow{G})$, is defined by

$$\epsilon(\overrightarrow{G}) = \max_{u \in \mathscr{V}} \frac{C_b(\overrightarrow{T}_u^{\overrightarrow{G}})}{C_b(T_u^{\overrightarrow{G}_{\max}})},$$

where $\overrightarrow{T}_u^{\overrightarrow{G}}$ is any arborescence of $\overrightarrow{G}$, rooted at u, of minimal broadcast cost, and $\overrightarrow{T}_u^{\overrightarrow{G}_{\max}}$ is any arborescence of $\overrightarrow{G}_{\max}$, rooted at u, of minimal broadcast cost.

Finding the broadcast stretch factor involves finding an arborescence of the max-power graph. This problem was proved to be NP-hard in (Cagalj et al., 2002). Finding an arborescence is related to the RA problem **Broadcast**. In Theorem 6.2.19 we state that **Broadcast** is NP-hard. This result was formally proved in (Clemeniti et al., 2001) and in (Cagalj et al., 2002). Observe that we have $\beta > 1$. This is because for $0 < \beta \leq 1$, the optimal solution is just for the broadcasting node to have a range larger than the distance to the farthest node, and all other nodes have range zero.

Theorem 6.2.19. *Solving* **Broadcast** *for $n \in \{2\}$ and $\beta > 1$ is NP-hard.*

Since it is NP-hard to compute an arborescence of minimal broadcast cost, it is necessary to find algorithms to approximate the optimal solution. Several such algorithms have been presented and one such algorithm is the so called *Broadcast Incremental Power* (BIP) *algorithm* (see Algorithm 3) proposed in (Wieselthier et al., 2000). The BIP-algorithm is a modification of the famous Prim's algorithm for finding the MST.

For the BIP-algorithm we need Definition 6.2.20.

Definition 6.2.20. Let $N_p = (\mathscr{N}, p)$ be a WSN. Let r be a range assignment for a subset $\mathscr{N}'$ of $\mathscr{N}$, which induces a communication graph $H = \overrightarrow{G_r}(\mathscr{N}')$ on $\mathscr{N}'$. For any $u \in \mathscr{N} \setminus \mathscr{N}'$ let r_u be the range assignment for $\mathscr{N}'$ such that for any $v \in \mathscr{N}'$, $r_u(v) = \delta(p(u), p(v))$. Then the *incremental cost of u with respect to H*, denoted by $IC_H(u)$, is defined by

$$IC_H(u) = \min_{v \in \mathscr{N}'} (r_u(v) - r(v)).$$

Unfortunately, the BIP-algorithm is a centralized algorithm and requires that every node has global knowledge about the network.

6.3 Coverage Problems

Coverage problems refers to how well the area of interest is monitored. Sometimes it might not be necessary to use all the sensors in the network to

Algorithm 3: Broadcast Incremental Power (BIP) Algorithm

Data: $\mathscr{N}$ - set of nodes;
u - broadcasting node;
$\mathscr{C}$ - set of nodes that are covered so far;
T - spanning tree under construction;
Result: T - Arborescence rooted at u

Initialize;
$\mathscr{C} = \{u\}$;
$T = \{u\}$;
while $\mathscr{C} \neq \mathscr{N}$ **do**
  for $v \in \mathscr{N} \setminus \mathscr{C}$ **do**
    compute $IC_T(v)$;
  choose $v_{\min} \in \mathscr{N} \setminus \mathscr{C}$ such that for any $v \in \mathscr{N} \setminus \mathscr{C}$, $IC_T(v_{\min}) \leq IC_T(v)$;
  $\mathscr{C} = \mathscr{C} \cup \{v_{\min}\}$;
  add $v_{\min}$ to T resulting in extra cost $IC_T(v_{\min})$;

achieve full coverage, in which case it is more energy-efficient to switch off the radio of the redundant sensor nodes.

In a general case, the area that is monitored by a sensor node need not have the shape of a ball (3D) or a disc (2D). However, to simplify the situation, we will assume that this is the case and throughout this section we say that a point p is within the *sensing region* of a node n whenever the distance between p and n is less than or equal to the *sensing range* of n.

Definition 6.3.1. Given a WSN $N_p = (\mathscr{N}, p)$, a *sensing assignment* for N_p is a function $s : \mathscr{N} \to (0, s_{\max}] \subset \mathbb{R}$, assigning to every node $n \in \mathscr{N}$ a *sensing range* $s(n)$.

Definition 6.3.2. Given a WSN $N_p = (\mathscr{N}, p)$ such that $p(\mathscr{N}) \subset \mathbb{R}^d$, we say that a point $x \in \mathbb{R}^d$ is *covered* by $n \in \mathscr{N}$ if $\delta(x, p(n)) \leq s(n)$. We say that x is *k-covered* if it is covered by at least k distinct nodes.

Note that we also say that x is covered by N_p if $\delta(x, p(v)) \leq s(v)$ for any node $v \in \mathscr{N}$ and we may also say that x is covered by $p(n)$ when we mean that x is covered by n.

The coverage problem is typically divided into three parts:

1. **Full coverage:** *every point* in the area of interest is covered;

2. **Barrier coverage:** given a belt-like region, each *crossing path* is covered;

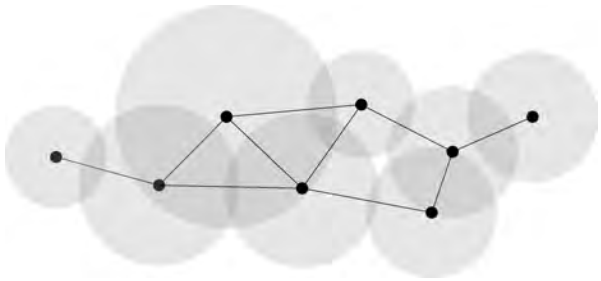


Figure 6.8 A Coverage Graph on a set of nodes.

3. **Sweep coverage:** mobile sensors cover the area/points of interest periodically.

Both full coverage and barrier coverage requires that the coverage criteria is satisfied all the time which we call *static coverage*, whereas sweep coverage is in the category of *dynamic coverage*.

Definition 6.3.3. Let $N_p = (\mathscr{N}, p)$ be a wireless sensor network, $p(\mathscr{N}) \subset \mathbb{R}^d$, $d \in \{1, 2, 3\}$ and let s be a sensing assignment for N_p. The subset of $\mathbb{R}^d$, covered by a node $n \in \mathscr{N}$ is called the *sensing region* of n and denoted by $\mathscr{D}_s(n)$. The boundary of $\mathscr{D}_s(n)$ will be denoted by $\mathcal{C}_s(n)$.

The choice of letter $\mathscr{D}$ becomes obvious when $d = 2$, where d is as in Definition 6.3.3. We will now define what we call the *coverage graph* and we do this in a slightly different manner than for the communication graph.

Definition 6.3.4. Let $N_p = (\mathscr{N}, p)$ be a WSN, $p(\mathscr{N}) = \mathscr{V}$ and s a sensing assignment for N_p. The *coverage graph* of N_p with respect to s, is the graph $G_s = (\mathscr{V}, \mathscr{E})$ such that for any two vertices $u, v \in \mathscr{V}$, $\{u, v\} \in \mathscr{E}$ if and only if $\mathscr{D}_s(u) \cap \mathscr{D}_s(v) \neq \emptyset$.

6.3.1 Full coverage

Full coverage means that every single point in the area of interest is covered by the network.

In this section, we review some of the work done in (Huang and Tseng, 2005).

Suppose that we want to cover a set $\mathscr{R} \subset \mathbb{R}^2$ by a set $\mathscr{N}$ of sensors with sensing range s. The sensing regions of $\mathscr{N}$ will divide $\mathscr{R}$ into a number of *subregions* and each subregion will be bounded by a number of circular segments, each belonging to some $\mathcal{C}_s(n)$, $n \in \mathscr{N}$.

Definition 6.3.5. Let $\mathscr{R} \subset \mathbb{R}^2$ and $N_p = (\mathscr{N}, p)$ a WSN such that $p(\mathscr{N}) \subset \mathscr{R}$. R *subregion* $\mathscr{R}'$ *of* $\mathscr{R}$ is a subset of $\mathscr{R}$ such that for any two points

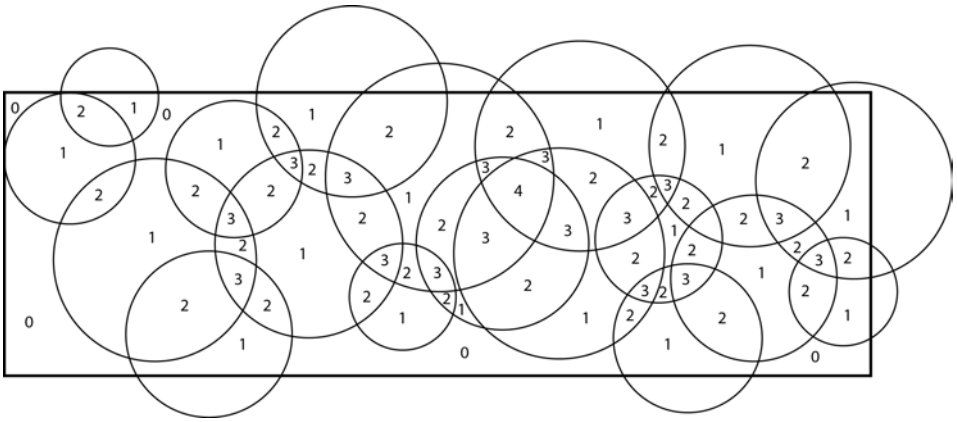


Figure 6.9 Subregions induced by a set of nodes. The number in each subregion tells how many nodes are covering the subregion.

$x, y \in \mathscr{R}'$ there exists a curve $l \subset \mathscr{R}$ between x and y such that for all $n \in \mathscr{N}$, $\mathcal{C}_s(n) \cap l = \emptyset$.

Definition 6.3.6. Let $n, n' \in \mathscr{N}$ be two nodes in a WSN $N_p = (\mathscr{N}, p)$ and s a sensing assignment for N_p. A point x in $\mathcal{C}_s(n)$ is said to be *perimeter-covered by* n' if $x \in \mathscr{D}_s(n')$.

Definition 6.3.7. A node n is said to be *k-perimeter-covered* if and only if every point in $\mathcal{C}_s(n) \cap \mathscr{R}$ is *perimeter-covered* by at least k distinct nodes other than n. Similarly, a segment $\mathscr{L}$ of $\mathcal{C}_s(n)$ is said to be *k-perimeter-covered* if and only if every point in $\mathscr{L}$ is *perimeter-covered* by at least k distinct nodes other than n. Here $\mathscr{R}$ is the area to be covered.

For Lemma 6.3.8 and Theorem 6.3.9, let $N_p = (\mathscr{N}, p)$, $p(\mathscr{N}) \subset \mathscr{R} \subset \mathbb{R}^2$ and let s be a sensing assignment for N_p.

Lemma 6.3.8. *Let $n \in \mathscr{N}$ and let $\mathscr{L}_n$ be a segment of $\mathcal{C}_s(n)$, which divides two subregions $\mathscr{R}_i, \mathscr{R}_o$ of $\mathscr{R}$, where $\mathscr{R}_i \subset \mathscr{D}_s(n)$ and $\mathscr{R}_o \not\subset \mathscr{D}_s(n)$. If $\mathscr{L}_n$ is k-perimeter-covered, then $\mathscr{R}_o$ is k-covered and $\mathscr{R}_i$ is $(k+1)$-covered.*

Theorem 6.3.9. *$\mathscr{R}$ is k-covered if and only if each $n \in \mathscr{N}$ is k-perimeter-covered.*

Proof. First we prove the 'if part'. Any subregion $\mathscr{R}_j$ in $\mathscr{R}$ is bounded by at least one segment of some $\mathcal{C}_s(n)$. Hence by Lemma 6.3.8, $\mathscr{R}_j$ is either $(k+1)$-covered or k-covered.

For the 'only if part', let $\mathscr{R}_i, \mathscr{R}_o$ be any two subregions divided by a segment $\mathscr{L}_n$ corresponding to a node $n \in \mathscr{N}$. Assume also that $\mathscr{R}_i \subset \mathscr{D}_s(n)$ and $\mathscr{R}_o \not\subset \mathscr{D}_s(n)$. Since $\mathscr{R}_o$ is k-covered and $\mathscr{R}_i$ is also covered by n, $\mathscr{R}_i$ must be $(k+1)$-covered. Hence each point in L_n is covered by k distinct nodes other than n. □

In (Huang and Tseng, 2005) Huang et. al. shows an algorithm which determines whether all nodes in $\mathscr{N}$ are perimeter covered or not and by Theorem 6.3.9 this also determines whether or not $\mathscr{R}$ is fully covered.

Coverage vs. Connectivity

A question that arises from the full coverage problem is whether there is a relation between coverage and connectivity. The following result (except for the part concerning $DG(\mathscr{V})$) is showed in (Wang et al., 2003).

Proposition 6.3.10. *Let $N_p = (\mathscr{N}, p)$ be a WSN with a homogeneous range assignment r, a homogeneous sensing assignment s and let $\mathscr{V} = p(\mathscr{N}) \subset \mathbb{R}^d$, $d \in \{1, 2, 3\}$. Suppose that N_p fully covers a convex region $\mathscr{R}$ and that $r \geq 2s$. Then the undirected communication graph $G_r = (\mathscr{V}, \mathscr{E})$ is connected. If $d = 2$ then $DG(\mathscr{V}) \subseteq G_r$, where $DG(\mathscr{V})$ is the Delaunay graph of $\mathscr{V}$.*

Proof. Suppose that $r \geq 2s$. Let $\mathscr{V}(v)$ denote the Voronoi region of a vertex v. First we will show that if $u, v \in \mathscr{V}$ and $\mathcal{V}(u) \cap \mathcal{V}(v) \neq \emptyset$, then $\{u, v\} \in \mathscr{E}$. Let x be any point in $\mathcal{V}(u) \cap \mathcal{V}(v)$. Then x is covered by both u and v. But this means that $\delta(u, v) \leq \delta(u, x) + \delta(x, v) \leq 2s \leq r$ and hence $\{u, v\} \in \mathscr{E}$.

Now let u', v' be any two vertices in $\mathscr{V}$. We must show that there exists a path from u' to v'. Let $\mathscr{L}$ be the straight line segment from u' to v' and suppose that $\mathscr{L}$ intersects the Voronoi regions $\mathcal{V}(u') = \mathcal{V}(v_1), \mathcal{V}(v_2), \ldots, \mathcal{V}(v_k) = \mathcal{V}(v')$ and that $\mathcal{V}(v_i) \cap \mathcal{V}(v_{i+1}) \neq \emptyset$, for every $1 \leq i \leq k-1$. Then the path $P = u'v_2v_3 \ldots v_{k-1}v'$ exists is G_r.

To show that $DG(\mathscr{V}) \subseteq G_r$, let $DG(\mathscr{V}) = (\mathscr{V}, \mathscr{E}')$ and let $\{u, v\}$ be any edge in $\mathscr{E}'$. Let $w \in \mathscr{V}$ such that the disc with u, v and w on its boundary contains no vertices in its interior. This implies that $\mathcal{V}(u) \cap \mathcal{V}(v) \cap \mathcal{V}(w) \neq \emptyset$ and hence, by the first part, $\{u, v\} \in \mathscr{E}$. □

Corollary 6.3.11. *Let $N_p = (\mathscr{N}, p)$ be a WSN with a range assignment r and a sensing assignment s. Suppose that $\mathscr{N}$ fully covers a convex region $\mathscr{R}$ and that for any two nodes u, v covering a common point, we have that $r(u) \geq s(u) + s(v)$ and $r(v) \geq s(u) + s(v)$. Then G_r is connected.*

6.3.2 Barrier coverage

Suppose that you are given the task to monitor a border to a forbidden area. To ensure detection of anyone crossing the border, you should place the sensors such that at least one point in each crossing path is covered by the "sensor barrier". Figure 6.10 illustrates this.

Barrier coverage has an obvious application in surveillance, when the goal is to detect any intruder crossing a border to a forbidden area. We will now show a model and some basic results showed by Kumar et. al in (Kumar et al., 2005).

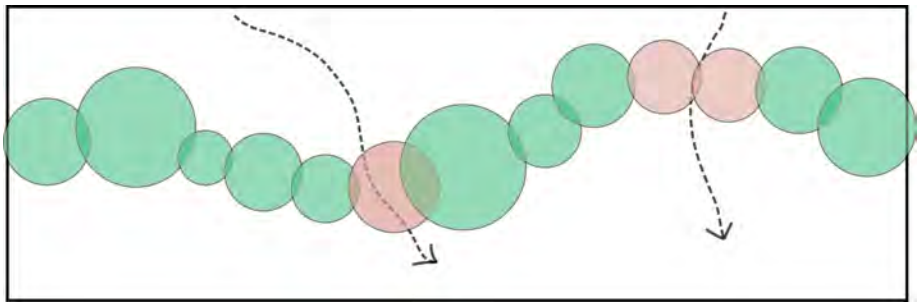


Figure 6.10 Barrier coverage

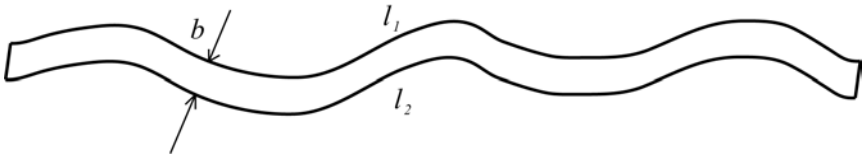


Figure 6.11 a belt $\mathscr{B}(l_1, l_2, b)$

Model

We will assume that the area of interest is a subset of $\mathbb{R}^2$ and that all sensors have a sensing area which has the shape of a disc. Furthermore, the sensing range s is the same for all nodes and hence we may refer to s as a real number.

Definition 6.3.12. A curve $l \subset \mathbb{R}^2$ is said to be *k-covered* by $\mathscr{N}$ if $l \cap \mathscr{D}_s(u) \neq \emptyset$ for at least k distinct nodes $u \in \mathscr{N}$.

Definition 6.3.13. If $l \subset \mathbb{R}^2$ is a curve and $p \in \mathbb{R}^2$, the *distance* from l to p, $\delta(l, p)$, is defined by

$$\delta(l, p) = \min_{x \in l} \delta(x, p)\,.$$

Definition 6.3.14. A set $\mathscr{B}(l_1, l_2, b) \subset \mathbb{R}^2$ is called a *belt* if it is bounded by two curves l_1, l_2 such that $\delta(l_1, y) = \delta(l_2, x) = b$ for all $x \in l_1$ and for all $y \in l_2$, for some $b \in \mathbb{R}$.

An example of a belt is a rectangle, which we will later define as an *open belt*. Another example of a belt is the set $\{\bar{x} \in \mathbb{R}^2 : 1 \leq |\bar{x}| \leq 2\}$, which we will define as a *closed belt*. The belt in Figure 6.12 is an open belt.

Definition 6.3.15. Let $\mathscr{B}(l_1, l_2, b) \subset \mathbb{R}^2$ be a belt. A *crossing path* $\gamma \subset \mathscr{B}(l_1, l_2, b)$, is a curve which starts in l_1 and ends in l_2.

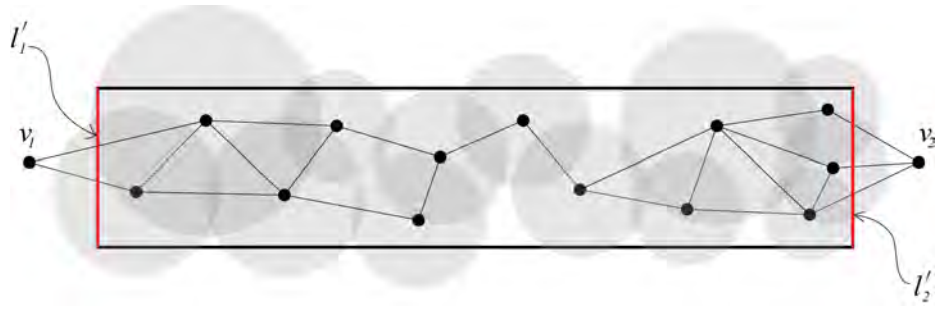


Figure 6.12 A barrier graph where v_1 and v_2 are 1-connected.

Definition 6.3.16. Let N_p be a WSN, $p(\mathscr{N}) \subset \mathbb{R}^2$ and $\mathscr{B} \subset \mathbb{R}^2$ a belt. $\mathscr{B}$ is said to be *k-covered* by $\mathscr{N}$ if and only if every crossing path in $\mathscr{B}$ is k-covered by $\mathscr{N}$.

Definition 6.3.17. A belt $\mathscr{B} \subset \mathbb{R}^2$ is called *open* if and only if for any two points $x, y \in \mathbb{R}^2 \setminus \mathscr{B}$, there is a curve l between x and y such that $\mathscr{B} \cap l = \emptyset$. A belt that is not open is said to be *closed*.

We will make a quite informal construction of a graph that can be used for the barrier coverage problem. Let $N_p = (\mathscr{N}, p)$ be a WSN with sensing range s and $p(\mathscr{N}) = \mathscr{V} \subset \mathscr{B} = \mathscr{B}(l_1, l_2, b)$ where $\mathscr{B}$ is an open belt. Let $G_s = (\mathscr{V}, \mathscr{E})$ be the coverage graph of N_p with respect to s. There are two straight lines of length b between endpoints of l_1 and l_2. We denote these two by l_1' and l_2'. Now choose two points v_1, v_2 not in $\mathscr{B}$ but close to l_1', l_2' respectively and for any $u \in \mathscr{V}$ add $\{u, v_i\}$ to $\mathscr{E}$ if and only if $\mathscr{D}_s(u) \cap l_i' \neq \emptyset$, $i \in \{1, 2\}$. Call this new edge-set $\mathscr{E}'$ and let $\mathscr{V}' = \mathscr{V} \cup \{v_1, v_2\}$. If v_1 and v_2 are such that any crossing path in $\mathscr{B}$ intersects any path from v_1 to v_2 in $G_{\max} = (\mathscr{V}', \mathscr{E}_{\max})$, where $\mathscr{E}_{\max}$ is the edge-set of the complete graph on $\mathscr{V}$, then we call $G' = (\mathscr{V}', \mathscr{E}')$ the *barrier graph* with respect to v_1, v_2.

Theorem 6.3.18. *Let $N_p = (\mathscr{N}, p)$ be a WSN with sensing range s and $\mathscr{V} = p(\mathscr{N}) \subset \mathscr{B}$ where $\mathscr{B}$ is an open belt such that the following condition holds: If $n, n' \in \mathscr{N}$ is any two nodes such that $\mathscr{D}_s(n) \cap \mathscr{D}_s(n') \neq \emptyset$, then $(\mathscr{D}_s(n) \cup \mathscr{D}_s(n')) \cap \mathscr{B}$ is a non-empty connected subset of $\mathscr{B}$.*
Then $\mathscr{B}$ is k-covered if and only if v_1 and v_2 are k-connected in the corresponding barrier graph $G = (\mathscr{V} \cup \{v_1, v_2\}, \mathscr{E})$.

Proof. For the 'if part', assume v_1 and v_2 are k-connected. By definition, there are k node-disjoint paths from v_1 to v_2. Hence, by construction of the barrier graph, each crossing path γ intersects k node-disjoint paths from v_1 to v_2 which means that there are at least k distinct nodes covering at least one point $x_k \in \gamma$ each. Hence, $\mathscr{B}$ is k-covered.

To prove 'only if', assume $\mathscr{B}$ is k-covered. To derive a contradiction, assume that v_1 and v_2 are not k-connected. Then by Menger's theorem

(Theorem E.1.26), there is a subset $W \subset \mathscr{V}$ of $(k-1)$ vertices which separates v_1 from v_2. Hence, by removing W from $\mathscr{V}$, we can find a crossing path γ' that is not covered and by adding W to $\mathscr{V}$ again we see that γ' is at most $(k-1)$-covered. This is a contradiction since $\mathscr{B}$ was assumed to be k-covered. □

6.3.3 Sweep covarage

Full coverage and barrier coverage requires a static and continuous coverage all the time. In sweep coverage, mobile sensors are used to cover the points in the area of interest periodically.

The concept of sweep coverage originates in robotics and was introduced for WSNs in (Li et al., 2012). The model we will use in this section is the same as in (Li et al., 2012) and the points to be covered are called *points of interest* (POIs). Every POI is visited at a certain time interval τ which guaranties event detection within a delay bound corresponding to τ.

Model

As before we will denote a WSN by $N_p = (\mathscr{N}, p)$ where $p(\mathscr{N}) \subset \mathbb{R}^d$, $d \in \{1, 2, 3\}$. The difference is that the nodes are now mobile and hence p also depends on the time t, i.e. $p(\mathscr{N}) = p(\mathscr{N}, t)$. We will denote a set of POIs by $P \subset \mathbb{R}^d$ and P will always be finite. For a given set P of POIs, we will associate a function $\tau : P \to [0, \infty) \subset \mathbb{R}$ which tells how often each POI should be visited. We call τ a *coverage time-interval*.

Definition 6.3.19. Let $N_p = (\mathscr{N}, p(t))$ be a mobile WSN, P a set of POIs and $\tau : P \to [0, \infty) \subset \mathbb{R}$, a coverage time-interval. Suppose that the WSN is set up to monitor the POIs during the time interval $[0, t_{end}]$. A POI $x \in P$ is said to be *sweep covered* by N_p if and only if for every $t \in [0, t_{end} - \tau(x)]$ there is a $t' \in [t, t + \tau(x)]$ and an $n \in \mathscr{N}$ such that $p(n, t') = x$.

In other words, Definition 6.3.19 says that p is sweep covered by n if and only if the node/sensor n is at the position p at least once every τ time unit.

Suppose know that we are given a set $\mathscr{N}$ of nodes and want to find a position function $p(t)$ for $\mathscr{N}$ such that P is sweep covered by $N_p = (\mathscr{N}, p(t))$. Suppose that all the nodes move with the same given speed v. Then we may ask for the minimum number of nodes $|\mathscr{N}|$, for which there exists a function $p(t)$ such that P is sweep covered by $N_p = (\mathscr{N}, p(t))$. The following theorem was proved in (Li et al., 2012).

Theorem 6.3.20. *Let $P = \{x_1, x_2, \ldots, x_n\}$ be a set of POIs, τ_i a coverage time-interval for each POI and v the speed of sensors. The problem of determining the minimum number $|\mathscr{N}|$ of sensors $\mathscr{N}$, for which there exists a position function $p(t)$ such that P is sweep covered by $N_p = (\mathscr{N}, p(t))$, is NP-hard.*

Proof. We will show a reduction from the TRAVELING SALESMAN PROBLEM (TSP) stated as follows:
INSTANCE: A finite set $C = \{c_1, c_2, \ldots, c_n\} \subset \mathbb{R}^2$ of "cities" and a constant $L \in \mathbb{R}^+$.
QUESTION: Is there an ordering $c_{\sigma(1)}, c_{\sigma(2)}, \ldots, c_{\sigma(n)}$ of the cities in C such that

$$\delta(c_{\sigma(n)}, c_{\sigma(1)}) + \sum_{i=1}^{n-1} \delta(c_{\sigma(i)}, c_{\sigma(i+1)}) \leq L?$$

Take any instance of TSP and let the cities be POIs to be covered. For every $c_i \in C$ set $\tau_i = L/v$. Let m be the least number of sensors needed to sweep cover C. Now it is easy to see that the TSP instance has a "yes answer" if and only if $m = 1$. Hence, if we find m, we solve the TSP-problem. □

6.4 Contents

In Section 6.1, some of the material is based on (Santi, 2005). Section 1.1.4 is taken from (Mao et al., 2012), (Mo et al., 2009) and (Li et al., 2013). Section 6.2.1 is mainly based on (Fuchs, 2006). Some material is based on (Kirousis et al., 2000) and (Garey and Johnson, 1979). Section 6.2.2 is based on (Santi, 2005), chapter 8. Section 6.3: the choice of content is similar to the coverage section in (Li et al., 2013), and the material is mainly based on three articles. Section 6.3.1 is based on (Huang and Tseng, 2005) and the last part from (Wang et al., 2003). Section 6.3.2 is based on (Kumar et al., 2005) and Section 6.3.1 introduces the work made in (Li et al., 2012). The first part of the appendix on Graph Theory is mainly based on (Diestel, 2005) and the section about proximity graphs is based on (de Berg et al., 2008).

6.5 Exercises

Problem 6.1 Connectivity

(a) Given a set $\mathscr{N}$ of nodes at positions

$$\mathscr{V} = \{(4,3), (6,2), (1,4), (10,2), (11,7), (9,1), (5,6), (7,3), (6,4), (2,6), (9,6)\},$$

use the MST-approximation algorithm and find $r_T(u)$ for every node $n \in \mathscr{N}$. Furthermore, compute the cost of r_T.
Hint: A MST is constructed by time after time adding the shortest edge connecting disconnected components (the Greedy algorithm).

(b) Prove whether or not the cost of r_T depends on what minimal spanning tree T you choose.

(c) Suppose that a set of nodes are randomly distributed in a region. Suppose that we are given a connected topology that minimizes the maximum vertex degree in the induced communication graph $G = (\mathscr{V}, \mathscr{E})$. What is the highest possible value of this maximum vertex degree? (i.e. a worst-case scenario).

Problem 6.2 Proximity Graphs

Consider the set $\mathscr{V} = \{(2,6),(5,3),(6,4),(7,3),(9,6),(10,2)\}$.

(a) Find the set of edges in GG, RNG, $EMST$ and DG.

(b) Check that $EMST \subseteq RNG \subseteq GG \subseteq DG$ holds.

(c) Assuming that each node knows its position, propose an algorithm to construct the $RNG(G_{\max})$ on a set $\mathscr{N}$ of nodes.

Problem 6.3 Coverage

(a) Let $\mathscr{B} \subset \mathbb{R}^2$ be a rectangular belt and $N_p = (\mathscr{N}, p)$ a WSN deployed in $\mathscr{B}$, i.e. $p(\mathscr{N}) = \mathscr{V} \subset \mathscr{B}$. Let $G = (\mathscr{V} \cup \{v_1, v_2\}, \mathscr{E})$ be a barrier graph ($v_1, v_2 \notin \mathscr{B}$). Suppose that k is the minimum number of vertices in $\mathscr{V}$ that have to removed from $\mathscr{V}$ to disconnect v_1 from v_2. Show that $\mathscr{B}$ is k-covered.

(b) Prove Corollary 6.3.11.

Problem 6.4 Algorithms for Proximity Graphs

Suppose that a set $\mathscr{N}$ of nodes are distributed in a plane area. Each node is equipped with a high precision directional antenna such that, if a node u transmits a message that reaches a node v, v can tell exactly from what direction the message was sent. Suppose further that every node $v_1 \in \mathscr{N}$ has a node $v_2 \in \mathscr{N}$ within its transmitting range. No node knows it's coordinate. Consider whether this information enough to determine the edge-set of the following cases:

(a) $RNG(G_{\max})$;

(b) $GG(G_{\max})$;

(c) $YG_6(G_{\max})$.

For (a), (b), and (c), explain how you concluded the answer and if the answer is affirmative do the following: for a given node $v \in \mathscr{N}$, propose an algorithm to determine the edges at v (the algorithm need not be fast).

Problem 6.5 NP-hardness

We formulate a decision problem called **ARBORESCENCE** as follows:

INSTANCE: A directed graph $G = (\mathscr{V}, \mathscr{A})$, a root vertex $u \in \mathscr{V}$ and $n \in \mathbb{N}$.
QUESTION: Does G contain an arborescence $T = (\mathscr{V}, \mathscr{A}')$ rooted at u and such that $|\mathscr{A}'| \leq n$?

A problem that is known to be NP-hard is **SET COVER**:

INSTANCE: A collection C of subsets of a finite set S and a positive integer $K \leq |C|$.
QUESTION: Is there a sub-collection $C' \subseteq C$ such that every element of S is contained in some member of C' and such that $|C'| \leq K$.

SET COVER can be represented by a directed bipartite graph $G_{SC} = (\mathscr{V}_{SC}, \mathscr{A}_{SC})$ with vertex-set $\mathscr{V}_{SC} = C \cup S$. The arc-set $\mathscr{A}_{SC}$ of G_{SC} is constructed as follows: for any $S' \in C$ and any $v \in S$, $(S', v) \in \mathscr{A}_{SC}$ if and only if $v \in S'$.

Use the graph G_{SC} to show that **ARBORESCENCE** is NP-hard.
Hint: add a root vertex u to $\mathscr{V}_{SC}$ and connect it in a smart way. Let $\{G = (\mathscr{V}_{SC} \cup \{u\}, \mathscr{A}_{SC}), u, n = K + |C|\}$ be an instance for **ARBORESCENCE**.

Chapter 7

Distributed Detection

In WSNs, one of the most important tasks is detecting correctly some event that is being sensed. There is a large variety of problems where we need to detect events, e.g., checking whether a specific event happened or not, presence or absence of noise is some measurements, whether delays are shorter or longer than a threshold, whether there have been collisions among messages, just to mention a few. On of the main challenges is that noise is often present in the sensor measurements, which may significantly alter them and give mistaken data interpretation.

Detection theory has been substantially developed in the last years so as to cope with this uncertainties, both in centralized settings and in distributed and networked setting. Recently, the theory has been extended to WSNs. Compared to classic detection theory, in WSNs there are more challenges since some sensor nodes are characterized by resource limitations, e.g., battery life, low transmission power, low processing capacity. To design an efficient WSN detection systems in WSNs, it is imperative to understand the essential detection theoretical results. This will allow to characterize the interplay between data compression, resource allocation, and overall performance in WSNs.

7.1 Basic Theory of Detection

The detection of events in WSNs is normally done by measuring signals. In this context, detection theory allow to quantify the ability to discern between information-bearing energy patterns of the signals, e.g. electrical signals, and random energy patterns that correspond to source of noise. Detection theory is a statistical technique to locate a signal against a background of noise.

It is important, to correctly detect signals, to distinguish between some finite number of possibilities of the signal's values and then define an algorithm that makes the best decision between these alternatives. This algorithm has

(a)Binary Hypotheses (b)Multiple Hypotheses

Figure 7.1 Hypothesis spaces: binary (left), and multiple (right).

to cope with the presence of uncertainties, such as white noise, based on some hypothesis from any prior knowledge, e.g., probability of some value of the signal, the mean value and the variance of the probability density function, among others.

The simplest instance of detection theory is binary hypothesis testing. Here, there are two hypothesis for detection, whether or not a signal is present. This is also called binary case of hypothesis. A more complex instance of detection theory is when multiple hypotheses are allowed and one has to decide which one among them is true. The goal of detection theory is to provide analytical tools to minimize the probability of error in the decision process. In what follows, we give a formalization of the basic theoretical results.

Consider the measurement $y(t)$ from a sensor, which is a noisy signal associated to an event we wish to detect. We assume that $y(t)$ is the outcome of a random variable, where the randomness comes from the measurement error. Such an error may be induced by the noises in the measurement devices or uncertainties about the phenomena. For example, when a node does a clear channel assessment in IEEE 802.15.4 MAC protocols, there is an uncertainty as to whether the channel is busy or free due to the random nature of the wireless propagation. Assuming $y(t) \in \mathbb{R}^2$, the cases of binary and multiple hypotheses are shown in Figure 7.1.

7.2 Detection from Single Sensor in Additive Noise

In this section, we study the case of detection from one single sensor. Later, we consider a WSN.

A sensor measures a signal $y(t)$ at time t from a physical phenomenon. In absence of any error in the measurement, the phenomenon gives a signal of value either 0 or μ_y. Consequently, there are two hypotheses:

$$\begin{cases} H_0 \text{ for hypothesis 0 } (S_0 \text{ signal}) \\ H_1 \text{ for hypothesis 1 } (S_1 \text{ signal}) \,. \end{cases} \tag{7.1}$$

However, the measurement is perturbed by additive white Gaussian error of zero mean and variance σ^2. It follows that $y(t)$ is modeled as

$$y(t) = \begin{cases} n(t) & \text{if } H_0 \text{ happened} \\ \mu + n(t) & \text{if } H_1 \text{ happened}\,. \end{cases} \tag{7.2}$$

The question is this: if we measure $y(t)$, how can we decide whether H_0 or H_1 happened?

In order to decide between the two hypotheses, there are several criteria available. Before proceeding to their formalization, we need to define some important probabilities. Let y be a realization of $y(t)$, the probability that H_0 or H_1 happened is called **a posteriori probability**. It is denoted by $\Pr(H_0|y)/\Pr(H_1|y)$. If the system detects correctly the signal, we define the **probability of detection**, $\Pr(s_1|H_1) = P_D$. If there is no signal and we obtain a positive observation, then it means it was a **false alarm**, with probability $\Pr(s_1|H_0) = P_F$. On the other hand, if there is a signal and it was not observed, then we have **probability of miss-detection**, $\Pr(s_0|H_1) = P_M$. These probabilities will be very useful to define the criterion to decide which hypothesis happened.

Several criteria may be defined so as to detect the value of a signal. Let us start with the maximum a posteriori (MAP) criterion. By MAP, the hypothesis is decided based on the highest conditional probability (given the observations). Specifically, in this criteria, the MAP probability is maximized after a set of measurements. In the case of binary hypothesis, the criterion consists in deriving a decision if a value is greater than or less than a previously defined threshold, namely, in MAP we decide for

$$\begin{cases} H_0 \text{ if } \Pr(H_0|y) > \Pr(H_1|y) \\ H_1 \text{ if } \Pr(H_1|y) > \Pr(H_0|y)\,, \end{cases} \tag{7.3}$$

and by using Bayes' rule we obtain

$$\Pr(H_0|y) = \frac{\Pr(y|H_0)\Pr(H_0)}{\Pr(y)} \tag{7.4}$$

$$\Pr(H_1|y) = \frac{\Pr(y|H_1)\Pr(H_1)}{\Pr(y)}\,. \tag{7.5}$$

Therefore MAP criterion is defined as,

$$\begin{cases} H_0 & \text{if } \Pr\left(y|H_0\right) \cdot \Pr\left(H_0\right) > \Pr\left(y|H_1\right) \cdot \Pr\left(H_1\right) \\ H_1 & \text{if } \Pr\left(y|H_0\right) \cdot \Pr\left(H_0\right) \leq \Pr\left(y|H_1\right) \cdot \Pr\left(H_1\right)\,. \end{cases} \tag{7.6}$$

The inequalities in (7.6) can be rearranged so to define the Likelihood Ratio Test (LRT)

$$\frac{\Pr\left(y|H_1\right)}{\Pr\left(y|H_0\right)} \gtrless \frac{\Pr\left(H_0\right)}{\Pr\left(H_1\right)}\,. \tag{7.7}$$

MAP and LRT are therefore identical. In general, let the probabilities of H_0 and H_1 be p_0 and p_1, respectively. Then the LRT tests is

$$\frac{\Pr(y|H_1)}{\Pr(y|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{p_0}{p_1} . \tag{7.8}$$

However, MAP take a special name, Maximum Likelihood (ML), when H_0 and H_1 are equally probable, i.e., $\Pr(H_0) = \Pr(H_1) = 1/2$.

If $\dfrac{\Pr(H_0)}{\Pr(H_1)} = 1 \quad \Rightarrow \quad$ Maximum likelihood detection (ML)

Under the assumption that the probabilities of H_0 and H_1, namely p_0 and p_1, are known beforehand, and recalling that the phenomenon is associated to the two signals, 0 and μ, with a Gaussian measurement error, we have

$$\Pr(y|H_0) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y-0)^2}{2\sigma^2}} \qquad \Pr(y|H_1) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y-\mu_y)^2}{2\sigma^2}} \tag{7.9}$$

and the LRT is

$$p_1 \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y-\mu_y)^2}{2\sigma^2}} > p_0 \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-y^2}{2\sigma^2}} , \tag{7.10}$$

which is equivalent to check if

$$y > \gamma = \frac{\mu_y^2 - 2\sigma^2 \ln\left(\frac{p_1}{p_0}\right)}{2\mu_y} . \tag{7.11}$$

Here we see that LRT reduced so checking is $y(t)$ is larger or smaller than a threshold γ. Based on this, we can define the probability of false alarm and miss detection as follows. The probability of false alarm and that of miss-detection are

$$\mathrm{P}_F(\gamma) \triangleq \Pr(H_1|H_0) = \int_{\gamma}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz = Q(\gamma) , \tag{7.12}$$

$$\mathrm{P}_M(\gamma) \triangleq \Pr(H_0|H_1) = \int_{-\infty}^{\gamma} \frac{1}{\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2}} dz = Q(\mu - \gamma) . \tag{7.13}$$

In these last two equations, we see that for a γ given by Equation (7.11), the probability of false alarm and of miss detection assume specific values. These values might not be adequate for some applications, where for example we may tolerate a high probability of false alarm, but we need to have a low

probability of miss detection. One natural question is, instead, to choose a different γ from Equation (7.11), so to have another desired false alarm and of miss detection probabilities.

To answer the question above, it can be noticed that both P_M and P_F change with γ, but if we try to decrease one of these two probabilities the other increases, namely the probability of false alarm and of miss detection are in opposition when varying γ. This leads to an optimization based approach to select a desired γ. Different optimization techniques can be used for this purpose such as Fast-Lipschitz optimization, Pareto optimization, etc. One optimization approach is to search for γ that minimizes P_M such that P_F is lesser than or equal to a constant value $\bar{\mathrm{P}}_F$. This will allow to set a specific optimal value of the threshold γ^*:

$$\begin{aligned} &\min_{\gamma} \ \mathrm{P}_M(\gamma) \\ &\text{s.t.} \quad P_F(\gamma) \leq \bar{\mathrm{P}}_F \end{aligned} \tag{7.14}$$

Since $\mathrm{P}_M(\gamma)$ decreases with γ, and since $P_F(\gamma)$ increases with γ, we deadly obtain the solution to this optimization problem γ^* as the solution to the following equation

$$Q(\gamma^*) = \bar{\mathrm{P}}_F \,. \tag{7.15}$$

We conclude this section with some observations. Both MAP and ML criteria can be extended to a multi-hypotheses situation by choosing the most likely of the M hypotheses, given the observations. Another useful criterion in detection is to try minimize the Bayes risk function, where the cost of false detection and miss is minimized and thus the results are more reliable. ML criterion which is used when prior probabilities are unknown can give similar decision probabilities as of MAP at high SNR. So it can be concluded that both criteria yield same results at high SNR while substantial difference in performance can be observed at low SNR.

7.3 Detection from Multiple Sensors

In a typical WSN scenario, several low-power and low-cost sensors are deployed to collect measurements and observations. The sensor nodes may pre-process or may not the measurements. These measurements (pre-processed or not) are then communicated to other nodes over the network to have a detection of the phenomena. For example, in a star network, there is a fusion center or cluster head that is responsible for processing information arriving from sensor nodes and making a global decision about the measured phenomena.

There are two alternatives for detection with multiple sensors over a star network. In the first approach, complete measurements or observations are

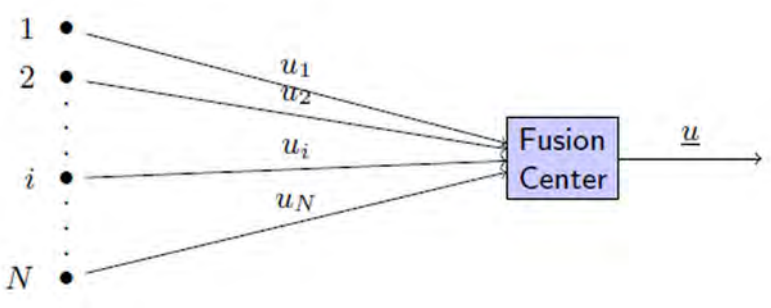


Figure 7.2 A distributed detection system with a data fusion center.

sent to the data fusion centre which processes the received messages to take a decision. This is called **centralized** detection system, which may require large communication bandwidth and low delay for transmission. The second alternative is to carry out all the signal processing at the sensor node level and to compute locally the result of the decision process. These results are then communicated to the fusion center which takes global decision based on the local decisions. This is known as **distributed** detection system. Such a distributed detection requires less communication bandwidth, which makes it useful for many WSNs applications. However, it may have the drawback of being less accurate as compared to sending all the measurements to a fusion center. A distributed system with data fusion is shown in Figure 7.2.

Let us now find an optimal detection system in a WSNs (Chair and Varshney, 1986) with a star topology where each node takes first its own decision on the phenomenon and then transmit such a decision to a fusion center. With this goal in mind, let us consider some preliminaries. Consider the binary hypothesis testing problem as defined in the previous section(H_0 means signal is absent, H_1 means signal is present). The a priori probabilities of the two hypotheses are denoted by $\Pr(H_0) = \mathrm{P}_0$ and $\Pr(H_1) = \mathrm{P}_1$. Assume that there are n sensor nodes in the star network, then the measurements from these sensors are denoted by $\mathbf{y}_i, i = 1, ..., n$. We further assume that the measurements at the individual detectors are statistically independent from each other, and that the conditional probability density function is denoted by $p(y_i|H_j), i = 1, ..., n, j = 1, 2$. Coherently with the results of the previous section, each sensor employs a decision rule $g_i(\mathbf{y}_i)$ to make a decision $u_i, i = 1, ..., n$, where

$$u_i = \begin{cases} 0, & \text{if } H_0 \text{ is detected} \\ 1, & \text{if } H_1 \text{ is detected}. \end{cases} \tag{7.16}$$

Also denote the probabilities of miss-detection and false alarms of each sensor by P_{Mi} and P_{Fi}, respectively. Recalling that each sensor in distributed detection system process its measurements, make local decision u_i and then

transmit its decision to the data fusion center, this saves communication bits, as opposed to sending many bits associated to the quantization of the measured data. Based on the local decisions, the data fusion center calculates the global decision, i.e.,

$$u = f(u_1, \ldots, u_n)\,. \tag{7.17}$$

Now we will consider the optimization of the data fusion algorithm.

Data fusion rules are often based on counting and are implemented as "k out of n logical functions. This means that if k or more detectors decide hypothesis H_1, then the global decision is H_1, i.e.,

$$u = \begin{cases} 1, & \text{if} u_1 + u_2 + \cdots + u_n \geq 2k - n \\ 0, & \text{otherwise}\,. \end{cases} \tag{7.18}$$

The data fusion problem can be viewed as a two-hypothesis detection problem where the decisions from the individual detectors $\mathbf{u}$ serves as the "measurement". The optimum decision rule is given by the following likelihood test:

$$\frac{\Pr(u_1, ..., u_n|H_1)}{\Pr(u_1, ..., u_n|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{\mathrm{P}_0(C_{10} - C_{00})}{\mathrm{P}_1(C_{01} - C_{11})}\,. \tag{7.19}$$

The left-hand side of the equation shows the likelihood ratio while the quantity on the right-hand is Bayes optimum threshold. If we assume the minimum probability of error criterion, that is, $C_{00} = C_{11}$=0, and $C_{10} = C_{01}$=1. Therefore we can write

$$\frac{\Pr(\mathbf{u}|H_1)}{\Pr(\mathbf{u}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{\mathrm{P}_0}{\mathrm{P}_1}\,. \tag{7.20}$$

The above equation can be simplified by using Bayes rule,

$$\frac{\Pr(\mathbf{u}|H_1)}{\Pr(\mathbf{u}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} 1\,. \tag{7.21}$$

The corresponding log-likelihood ratio test is

$$\log \frac{\Pr(\mathbf{u}|H_1)}{\Pr(\mathbf{u}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} 0\,. \tag{7.22}$$

An expression for implementation of the data fusion rule can be established and the result is presented by the following theorem.

THEOREM. *Given n detectors and the corresponding quantities as defined previously, we have*

$$\log \frac{\Pr(H_1|\mathbf{u})}{\Pr(H_0|\mathbf{u})} = \log \frac{\mathrm{P}_1}{\mathrm{P}_0} + \sum_{S_+} \log \frac{1 - \mathrm{P}_{Mi}}{\mathrm{P}_{Fi}} + \sum_{S_-} \log \frac{\mathrm{P}_{Mi}}{1 - \mathrm{P}_{Fi}}\,. \tag{7.23}$$

Problems

Problem 7.1 Binary choice in Gaussian noise

A signal voltage z can be zero (hypothesis H_0) or k (hypothesis H_1), each hypothesis with a probability 1/2, The voltage measurement is perturbed by additive white Gaussian noise (AWGN) of variance σ^2. Compute the decision threshold for MAP criterion, and the error probabilities $\Pr(D_1|H_0)$ and $\Pr(D_0|H_1)$, where D_1 means that H_1 was decided, and D_0 means H_0 was decided.

Problem 7.2 Binary hypothesis test and SNR (Ex.5.2 in (Pottie and Kaiser, 2005))

Consider the binary choice in Gaussian noise, as shown in previous exercise with the threshold of $k/2$, the SNR is also maximized at the decision point. Since the possible signal values are known, the maximization of SNR means that the hypothesized noise power $E[n^2(t)]$ is minimized when the decision boundaries are optimally chosen. Prove that SNR is maximized when the threshold is $k/2$.

Problem 7.3 MAP and the LRT (Ex.5.4 in (Pottie and Kaiser, 2005))

Show that the MAP decision rule is equivalent to the likelihood ratio test.

Problem 7.4 Binary decisions with unequal a priori probabilities(Ex.5.5 in (Pottie and Kaiser, 2005))

For the binary choice in Gaussian noise in Exercise 1, compute the threshold when the probabilities of H_0 and H_1 are 1/3 and 2/3 respectively.

Problem 7.5 Detection of known mean in Gaussian noise (Example D.1 in (Gustafsson, 2012))

The simplest possible problem is to decide whether there is a known mean A in an observed signal or not:

$$\begin{aligned} H_0 &: y_k = e_k \,, \\ H_1 &: y_k = s_k + e_k \,. \end{aligned}$$

Suppose to detect a general known signal s_k observed with Gaussian noise as $y_k = s_k + e_k$. Using a matched filter defined as

$$\bar{y} = \frac{1}{N} \sum_{k=0}^{N-1} y_k s_k = A + \bar{e} \,,$$

show that

$$A = \frac{1}{N} \sum_{k=0}^{N-1} s_k^2$$

and $\bar{e} \sim \mathcal{N}(0, \sigma^2/N)$. Here we assume that $\sum s_k = 1$.

Problem 7.6 Fault detection

Suppose to detect a signal s_k observed with Gaussian noise as $y_k = s_k + e_k$, where $e_k \sim \mathcal{N}(0, \sigma^2)$. Assume there exist fault alarms for the signal, that is, the alarms occur when the measurement of signal beyond the interval $[-3\sigma, 3\sigma]$. Here assume that s_k equals 0 with $p_0 = 0.9$, and $t = 3\sigma$ as fault with $p_t = 0.1$. Find the probability of the correct fault alarms.

Problem 7.7 Optimal Data Fusion in Multiple Sensor Detection Systems (Chair and Varshney, 1986)

Let consider a binary hypothesis problem with the following two hypotheses: H_0 signal is absent, H_1 signal is present. The priori probabilities of the two hypotheses are denoted by $\Pr(H_0) = P_0$ and $\Pr(H_1) = P_1$. Assume that there are n detectors and the observations at each detector are denoted by $y_i, i = 1, \ldots, n$. Furthermore, assume that the observations are statistically independent and that the conditional probability density function is denoted by $p(y_i|H_j), i = 1, \ldots, n$, while $j = 1, 2$. Each detector employs a decision rule $g_i(y_i)$ to make a decision $u_i, i = 1, \ldots, n$, where

$$u_i = \begin{cases} -1 & \text{if } \mathrm{H}_0 \text{ is declared} \\ +1 & \text{if } \mathrm{H}_1 \text{ is declared} \end{cases} .$$

We denote the probabilities of the false alarm and miss of each detector by P_{F_i} and P_{M_i} respectively. After processing the observations locally, the decisions u_i are transmitted to the data fusion center. The data fusion center determines the overall decision for the system u based on the individual decisions, i.e., $u = f(u_1, \ldots, u_n)$.

1. Show that

$$\log \frac{\Pr(H_1|\mathbf{u})}{\Pr(H_0|\mathbf{u})} = \log \frac{P_1}{P_0} + \sum_{S_+} \log \frac{1 - P_{M_i}}{P_{F_i}} + \sum_{S_-} \log \frac{P_{M_i}}{1 - P_{F_i}} ,$$

   where S_+ is the set of all i such that $u_i = +1$ and S_- is the set of all i such that $u_i = -1$.

2. Find the optimum data fusion rule using likelihood ratio.

Problem 7.8 Counting Rule (Niu and Varshney, 2005)

Consider the same situation in Exercise 7. An alternative scheme would be that the fusion center counting the number of detections made by local sensors and then comparing it with a threshold T:

$$\Lambda = \sum_{S_+} u_i \underset{H_0}{\overset{H_1}{\gtrless}} T ,$$

which is called "counting rule". Now assume that each sensor has the same $P_{F_i} = P_f$ and $P_{M_i} = P_m$, find the probability of false alarm P_F and detection P_D at the fusion center level.

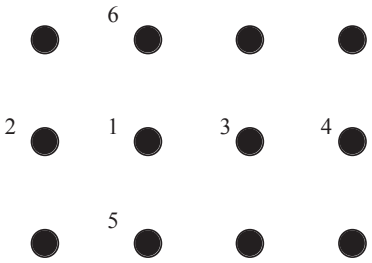


Figure 7.3 A grid of sensor nodes.

Problem 7.9 Distributed detection, MAC, and routing

Sensor nodes are laid out on a square grid of spacing d, as depicted in Figure 7.3. Ever sensor wants to detect a common source.

(a) Suppose that the source signal has a Gaussian distribution with zero mean and with variance σ_S^2. Moreover, every sensor measures such a signal with an additive Gaussian noise of zero average and variance σ_n^2. If the measured signal is positive, the sensor decides for hypothesis H_0, otherwise the sensor decides for hypothesis H_1. Based on the measured signal, characterize the probability of false alarm and the probability of miss detection per every sensor.

(b) Now, suppose that the source signal is constant and has a power S. Such a signal power is received at every sensor with an attenuation given by r_i^2, where r_i is the distance between the source and sensor i. Sensor node 1 is malfunctioning, producing noise variance $10\sigma_n^2$. The two best nodes in terms of SNR will cooperate to provide estimates of the source. Characterize the region of source locations over which node (1) will be among the two best nodes.

(c) The grid depicted in Figure 7.3 is also used for relaying. Assume it costs two times the energy of a hop among nearest neighbors (separated by distance d) to hop diagonally across the square (e.g. node 2 to 5) and eight times the energy to go a distance of 2d in one hop (e.g. node 2 to 3). Let p be the packet loss probability. Characterize p for which it is better to consider using two diagonal hops to move around the malfunctioning node.

(d) Under the same assumption of the previous item, suppose that there is an ARQ protocol, but the delay constraints are such that we can only tolerate three retransmission attempts. Let 0.99 be the probability of having up to three retransmissions. Assuming packing dropping events are independent, characterize the constraint that probability of packet losses per transmission should satifsy.

Problem 7.10 Matched filter and SNR (Ex.5.12 in (Pottie and Kaiser, 2005))

Prove the matched filter maximizes the output SNR and compute the maximum output SNR as a function of the energy of the signal $s(t)$ and N_0.

Problem 7.11 Binary hypothesis testing and mutual information (Ex.5.3 in (Pottie and Kaiser, 2005))

Consider the binary choice in Gaussian noise, as shown in Exercise 1. When $k = 1$ and the variance of the Gaussian distribution is 1, show numerically that the mutual information is maximized when $\gamma = 0.5$.

Chapter 8

Distributed Estimation

Distributed estimation plays an essential role in many networked applications, such as communication, networked control, monitoring and surveillance. Motivated by this, the chapter provides an overview on some of the fundamental aspects of distributed estimation over networks together with an investigation of the computational complexity and communication cost. A phenomenon being observed by a number of sensors in networks having a star and a general topology are considered. Under the assumptions of noises and linear measurements, the resulting distributed estimators are derived respectively. The limited bandwidth, communication range and message loss in the communication are considered. Distributed estimators can provide accurate estimates of the parameters of the phenomenon, while the less the limitations are in networks, the lower complexity of the estimator is.

8.1 Optimal Mean Square Estimate of a Random Variable

We will be interested in Minimum Mean Square Error (MMSE) estimates. Given a random variable Y that depends on another random variable X, obtain the estimate $\hat{X}$ such that the mean square error given by $\mathbb{E}\left[X-\hat{X}\right]^2$ is minimized. The expectation is taken over the random variables X and Y.

Proposition 8.1.1. *: The minimum mean square error estimate is given by the conditional expectation* $\mathbb{E}\left[X|Y=y\right]$.

Proof. The arguments are standard. Consider the functional form of the estimator as $g(Y)$. Let $f_{X,Y}(x,y)$ denote the joint probability density function of X and Y. Then the cost function C is given by

$$\begin{aligned}\mathbb{E}\left[X-\hat{X}\right]^2 &= \int_x\int_y (x-g(y))^2 f_{X,Y}(x,y)dxdy \\ &= \int_y dy f_Y(y)\int_x (x-g(y))^2 f_{X|Y}(x|y)dx.\end{aligned}$$

Now consider the derivative of the cost function with respect to the function $g(y)$.

$$\begin{aligned}\frac{\partial C}{\partial g(y)} &= \int_y dy f_Y(y) \int_x 2(x - g(y)) f_{X|Y}(x|y) dx \\ &= 2 \int_y dy f_Y(y)(g(y) - \int_x x f_{X|Y}(x|y) dx) \\ &= 2 \int_y dy f_Y(y)(g(y) - \mathbb{E}\left[X|Y = y\right]).\end{aligned}$$

Thus the only stationary point is $g(y) = \mathbb{E}\left[X|Y = y\right]$. Moreover it is easy to see that it is a minimum. □

The result holds for vector random variables as well.

MMSE estimates are important because for *Gaussian* variables, they coincide with the Maximum Likelihood (ML) estimates. Of course, for non-Gaussian random variables, other notions of optimality may be better.

It is also a standard result that for Gaussian variables, the MMSE estimate is linear in the state value. Proof was given in the lecture on Kalman filtering. So we will restrict our attention to linear estimates now. Also, from now on we will assume zero mean values for all the random variables. All the results can however be generalized. The covariance of X will be denoted by R_X and the cross-covariance between X and Y by R_{XY}.

Proposition 8.1.2. *The best linear MMSE estimate of X given $Y = y$ is*

$$\hat{x} = R_{XY} R_Y^{-1} y,$$

with the error covariance

$$P = R_X - R_{XY} R_Y^{-1} R_{YX}.$$

Proof. Let the estimate be $\hat{x} = Ky$. Then the error covariance is

$$\begin{aligned}C &= \mathbb{E}\left[(x - Ky)(x - Ky)^T\right] \\ &= R_X - K R_{YX} - R_{XY} K^T + K R_Y K^T.\end{aligned}$$

Differentiating C w.r.t. K and setting it equal to zero yields

$$-2R_{XY} + 2K R_Y^{-1} = 0.$$

The result follows immediately. □

In the standard control formulations, we are also interested in measurements that are related linearly to the variable being estimated (usually the state).

Proposition 8.1.3. *Let $y = Hx + v$, where H is a matrix and v is a zero mean Gaussian noise with covariance R_V independent of X. Then the MMSE estimate of X given $Y = y$ is*

$$\hat{x} = R_X H^T \left(HR_X H^T + R_V\right)^{-1} y,$$

with the corresponding error covariance

$$P = R_X - R_X H^T \left(HR_X H^T + R_V\right)^{-1} HR_X.$$

Proof. Follows immediately by evaluating the terms R_{XY} and R_Y and substituting in the result of Proposition 8.1.2. □

8.2 Network with a Star Topology

In this section, we assume that the network is organized as a star, where multiple sensors make measurements that are transmitted with no messages losses to a fusion center, which is assumed to be the star of the network. The fusion center is responsible for processing the data sent from all the sensors to arrive at an estimation of the measured quantity. Much of this section will focus on how to move as much as possible of the processing from the Fusion center out to the sensors doing the measurements. For this to be useful we have to assume that the sensors can do some computation, but that is a reasonable assumption.

An example of the star topology is illustrated by Fig. 8.1. Note that all the algorithms can be used for other connected topologies if routing of messages is possible. In that case every node that is not connected directly to the fusion center will still be able to send messages to the center with the help of interconnecting nodes.

8.2.1 Static Sensor Fusion

Here we study the problem of estimating a static phenomenon that is observed by a number of sensors. The observations of the sensors are reported to a central unit that fuses them with the aim of extracting an estimate of higher accuracy. Since the value that we are trying to estimate is constant in time each sensor will also be able to preform multiple measurements in order to get a better estimate. Thus the techniques are useful even when using only a single sensor. The main purpose of the procedures described in the section is to reduced the amount of processing required at the fusion center and reduce the amount of data that the sensors need to transmit. The approach to reduce the processing required at the fusion center is based on reducing the size of the matrices involved in the final estimation by using local estimates from the sensors instead of the actual measurements.

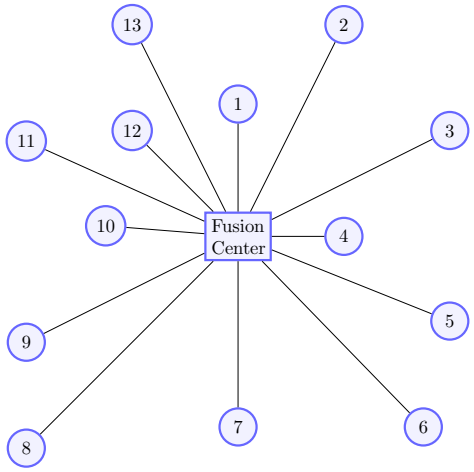


Figure 8.1 An example of star topology network with nodes and links (solid lines indicating that there is message communication between nodes). In this network, the fusion center can receive information from all other nodes.

Different estimators

When doing estimations there are many different approaches with different properties and it is important to choose a scheme that suits what the application needs. Some approaches put the focus on quick converges with a small amount of data but with a possibility of large errors while other approaches (such as the maximum likelihood estimator) can be inefficient for small data sets but becomes extremely accurate for large data sets. Choosing a different estimator based on the situation can thus improve the results significantly.

One of the most common ways to form estimates is the maximum likelihood estimator that is defined as

$$\hat{x} = \arg\max_{x} p(y|x). \tag{8.1}$$

This should be understood as for measurements y and a known (a prior) distribution $p(\cdot)$ the estimate of x should be chosen in such a way that the measurements y where the most likely given that $p(\cdot)$ has the expected value $\hat{x}$. Since the estimate is based on a distribution that is already known it is rather easy to include extra information about the variable that is being estimated. It is however not as easy to see how one would handle the case where there is no information about the prior distribution. Another disadvantage with Maximum likelihood estimation is that no simple formula for Distributed estimation applies for all distributions. To counter act this special tailored approaches are used for different common distributions and

numerical methods for the remaining cases. There are however some cases where a maximum of the likelihood function does not exist making the ML-estimator useless in those cases. The biggest advantage to the ML estimator is that it attains the lower bound on the variance asymptomatically with the size of the data set. (Gustafsson, 2012)

Based on this we will instead focus on another type of estimator in this chapter namely the Minimum Mean Square Error (MMSE) estimator that is very useful for distributed estimation. It also has the nice property that it is identical to the ML estimator for the common case of a Gaussian distribution so accuracy is guaranteed in such a case. We recall the results from centralized linear estimators from proposition **??** that we reiterate in proposition 8.2.1. We will use this form to get an expression that makes distributed estimation possible.

Proposition 8.2.1. *Let $y = Hx + v$, where H is a matrix and v is a zero mean Gaussian noise with covariance matrix R_V independent of X. Then the MMSE estimate of X given $Y = y$ is*

$$P^{-1}\hat{x} = H^T R_V^{-1} y,$$

with P as the corresponding error covariance given by

$$P^{-1} = \left(R_X^{-1} + H^T R_V^{-1} H\right).$$

For the above proposition P^{-1} will act as a normalizing factor to the weights formed by $H^T R_V^{-1}$. That P^{-1} acts as a normalizer will also be true for the distributed case later on. In the special case of no prior information the combination of the values will simply become a weighted least square problem for the equation $Hx = y$ which can be used to get a better feeling for how the system acts. Example 8.2.3 on page 174 also gives insight into how this representation behaves in the scalar case.

The fact that X is treated as a stochastic variable makes it possible to include extra information in the estimation even before any measurements have been done by including the available information in the probability distribution. An example on how such information effects the error is given in figure 8.2. The proposition above assumes that the expected value of X is 0. The estimate can however be generalised in the following way where $\bar{x}$ denotes the expected value of X

$$P^{-1}(\hat{x} - \bar{x}) = H^T R_V^{-1}(y - H\bar{x}).$$

For the rest of the chapter we will continue to work under the assumption that $\bar{x} = 0$ to save space since the generalization is quite straight forward. It is also worth noting that in the case where no information about X is available it is still treated as a stochastic variable and not a constant. This

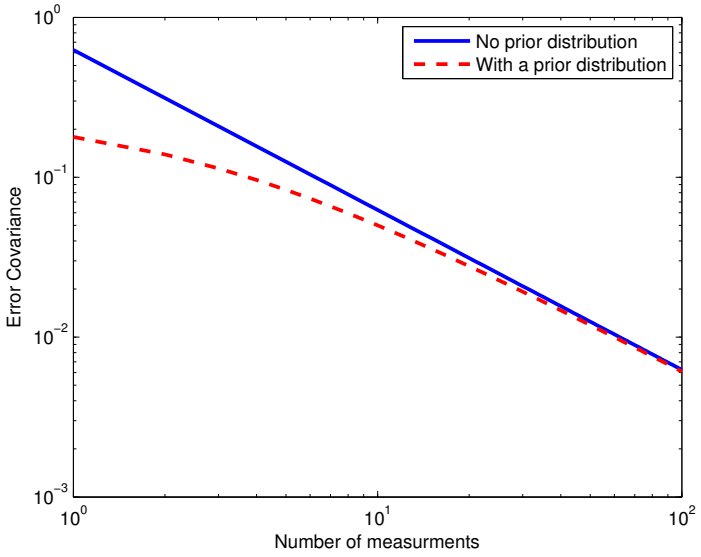


Figure 8.2 Effect on the error by including a prior distribution with 40 times lower variance than each measurement. As expected the effects of the prior is most significant with few measurements.

can be understood by thinking of the variance of X. If X is a constant the variance is 0 and we know what value it will attain, doing an estimation would thus be pointless. This is not the situation we want to model, in fact it is the exact opposite. We will thus in most cases have to consider X to have a variance that approaches infinity resulting in that R_X^{-1} goes towards 0.

We will now manipulate this alternate form to get a way to construct the global estimate at the fusion center from the local estimates at the sensors. This is called static sensor fusion.

Static Sensor Fusion for Star Topology

In this section we will show how a group of sensors can work together to get a better estimation of a time-invariant variable, the concept is illustrated in figure 8.3. The process distributes work normally required by the fusion center to the different sensors and then gathers the results in a way that has no loss of precision compared to the centralized version.

Proposition 8.2.2. *Consider a random variable x being observed by K sensors that generate measurements of the form*

$$y_k = H_k x + v_k, \qquad k = 1, \cdots, K,$$

where the noises v_k are all uncorrelated with each other and with the variable

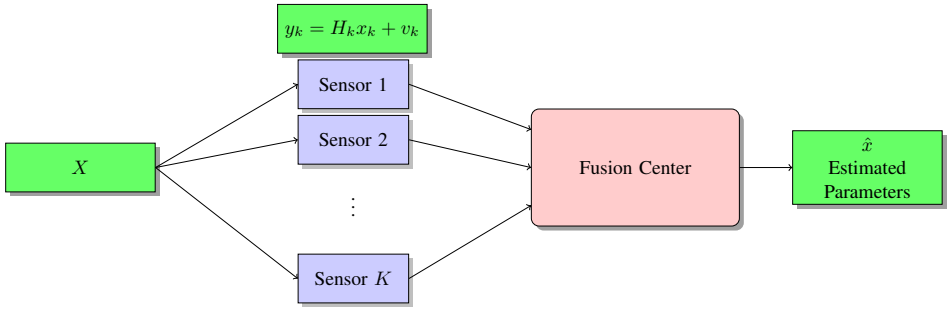


Figure 8.3 Illustration of the static fusion. Each sensor makes a noisy measurement of X. It then sends some quantity to the fusion center that forms the final estimate.

x. Denote the estimate of x based on all the n measurements by $\hat{x}$ and the estimate of x based only on the measurement y_k by $\hat{x}_k$. Then $\hat{x}$ can be calculated using

$$P^{-1}\hat{x} = \sum_{k=1}^{K} P_k^{-1}\hat{x}_k,$$

where P is the estimate error covariance corresponding to $\hat{x}$ and P_k is the error covariance corresponding to $\hat{x}_k$. Further

$$P^{-1} = \sum_{k=1}^{K} P_k^{-1} - (K-1)R_X^{-1}.$$

Proof. Denote y as the stacked vector of all the measurements y_k's, H the corresponding measurement matrix obtained by stacking all the H_k's in a block diagonal way and v the noise vector obtained by stacking all the noises v_k's. The global estimate $\hat{x}$ is given by

$$P^{-1}\hat{x} = H^T R_V^{-1} y.$$

But all the v_k's are uncorrelated with each other. Hence R_V is a block diagonal matrix with blocks R_{V_k}. Thus the right hand side can be decomposed as

$$H^T R_V^{-1} y = \sum_{k=1}^{K} H_k^T R_{V_k}^{-1} y_k.$$

But each of the terms $H_k^T R_{V_k}^{-1} y_k$ can be written in terms of the local estimates

$$P_k^{-1}\hat{x}_k = H_k^T R_{V_k}^{-1} y_k.$$

Thus

$$P^{-1}\hat{x} = \sum_{k=1}^{K} P_k^{-1}\hat{x}_k.$$

The proof for the expression for the global error covariance is similar. □

This result is useful since it allows the complexity of calculation at the fusion center to go down considerably[1]. The form of the global estimator shows that what we really want is a weighted mean of the local estimates. Each estimate is weighted by the inverse of the error covariance matrix. Thus the more confidence we have in a particular sensor, the more trust do we place in it. In the following example the concept is illustrated for a possible real case.

Example 8.2.3. *Suppose we have two sensors that measure the resistance of two different materials with the ultimate goal of measuring the temperature in the room. The measurements contains unbiased Gaussian noise v_1 and v_2 with the variances R_1 and R_2 respectively.*

The relation between the measured resistance and temperature is given by the equation:

$$R_i(T) = R_i(T_0)(1 + \alpha_i \Delta T) + v_i, \quad i = 1, 2.$$

What will the fused estimate be and how will the error change?

The equation for the temperature dependence can be manipulated in the following way

$$R_i(T) - R_i(T_0)(1 - T_0\alpha) = \alpha_i R_i(T_0)T + v_i$$

Since the measurement $R_i(T)$ and the constants are known we can with the following substitutions state this on the standard form of $y_i = H_i T + v_i$. The substitutions are

$$\begin{aligned} y_i &= R_i(T) - R_i(T_0)(1 - T_0\alpha), \\ H_i &= \alpha_i R_i(T_0). \end{aligned}$$

We can now use the result from proposition 8.2.2 to answer the questions. First we have for the error the following

$$\begin{aligned} P^{-1} &= P_1^{-1} + P_2^{-1} - (K-1)R_T^{-1} = P_1^{-1} + P_2^{-1} \\ &= H_1^2 R_1^{-1} + H_2^2 R_2^{-1} = \frac{H_1^2}{R_1} + \frac{H_2^2}{R_2}. \end{aligned}$$

The reason that R_T^{-1} vanishes is because we have no a priori information about T. Thus the variance of T can be considered to be infinite and R_T^{-1} becomes 0. Since the variance is scaler $H^T = H$ and H commutes, we can also get the explicit expression for the error covariance in the scalar case

$$P = \frac{R_2 R_1}{H_1^2 R_2 + H_2^2 R_1},$$

[1] As an exercise, compare the number of elementary operations (multiplications and additions) for the two algorithms.

this is obviously smaller than both P_1 and P_2.

The estimate is also given by proposition 8.2.2 and is

$$\hat{x} = \frac{R_2 R_1}{H_1^2 R_2 + H_2^2 R_1} \left(\frac{H_1}{R_1} y_1 + \frac{H_2}{R_2} y_2 \right).$$

Sequential Measurements from One Sensor

The same algorithm can be extended to the case when there are multiple measurements from one sensor in order to gain a higher accuracy in the estimate. Furthermore, the processing can be done in a sequential manner so the memory requirements are constant in time. This is essential for the algorithm to be useful since the memory available for each sensor is limited. Consider a random variable evolving in time as

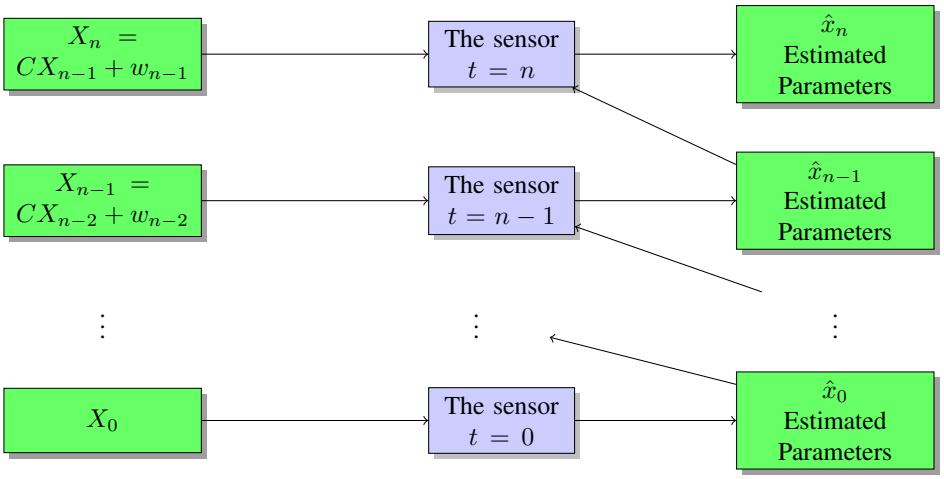


Figure 8.4 Illustration of fusion of multiple measurements in one sensor.

$$X_{n+1} = AX_n + w_n,$$

where w_n is white zero mean Gaussian noise with covariance matrix Q. The sensor generates a measurement at every time step according to the equation

$$Y_n = CX_n + v_n,$$

where v_n is again white zero mean Gaussian noise with covariance matrix R. We wish to obtain an estimate of X_n given all the measurements $\{Y_0, Y_1, \ldots, Y_n\}$. Suppose we divide the measurements into two sets:

1. The measurement Y_n.
2. The set $\mathcal{Y}$ of the remaining measurements Y_0 through Y_{n-1}.

Now note that the two sets of measurements are related linearly to X_n and further *the measurement noises are independent.* Thus we can combine the local estimates to obtain a global estimate. First we calculate the estimate of X_n based on Y_n in the same way as in the previous section. It is given by

$$M^{-1}\hat{X} = C^T R^{-1} Y_n,$$

where M is the error covariance given by

$$M^{-1} = R_{X_n}^{-1} + C^T R^{-1} C.$$

Let $\hat{X}_{n-1|n-1}$ be the estimate of X_{n-1} based on $\mathcal{Y}$ and $P_{n-1|n-1}$ be the corresponding error covariance. Then the estimate of X_n given $\mathcal{Y}$ is given by

$$\hat{X}_{n|n-1} = A\hat{X}_{n-1|n-1},$$

with the error covariance

$$P_{n|n-1} = AP_{n|n-1}A^T + Q.$$

Thus the estimate of X_n given all the measurements is given by the combination of local estimates and can be seen to be

$$P_{n|n}^{-1}\hat{X}_{n|n} = P_{n|n-1}^{-1}\hat{X}_{n|n-1} + M^{-1}\hat{X} = P_{n|n-1}^{-1}\hat{X}_{n|n-1} + C^T R^{-1} Y_n.$$

The corresponding error covariance is

$$P_{n|n}^{-1} = P_{n|n-1}^{-1} + M^{-1} - R_{X_n}^{-1} = P_{n|n-1}^{-1} + C^T R^{-1} C.$$

This can be seen as a fusion of measurements from two different sensors as seen in figure 8.4. The measurements in the set $\mathcal{Y}$ form a local estimate that is treated in the same way as if it originated from another sensor in the situation described in the previous case. The local estimate from $\mathcal{Y}$ is thus fused with the new measurement Y_n to from the global estimate.

These equations also form the time and measurement update steps of the Kalman filter. Thus the Kalman filter can be seen to be a combination of estimators. This also forms an alternative proof of the optimality of the Kalman filter in the minimum mean squared sense under the stated assumptions. We will give more detail on Kalman filtering, and in particular on distributed Kalman filtering below.

8.2.2 Dynamic Sensor Fusion

Suppose a combination of the previous two cases where there are multiple sensors present that generate measurements about a random variable that is evolving in time. We can again ask the question about how to fuse data from all the sensors for an estimate of the state X_n at every time step n.

This is the question of dynamic sensor fusion. It arises in many different situations, such as tracking rotations and observing osculating phenomena. A concrete example is temperature tracking at a shoreline, the tide will in such a situation oscillate in a very predictable way. This oscillation will cause the height of the water to change and thus effect the temperature of the water, the concept in illustrated in figure 8.5. We will now begin by seeing why this question is difficult in general.

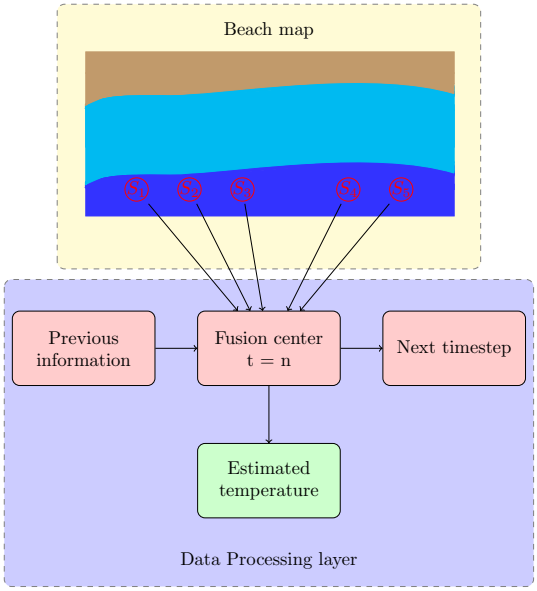


Figure 8.5 Illustration of measurements taken in a real world scenario, and how the data processing works. The cyan colored area represents land that can be covered with water during a high tide and S_i represents a sensor in the water.

To begin with, the problem can be solved if all the sensors transmit their measurements at every time step. The central node in that case implements a Kalman filter (which we will refer to from now as the *centralized* Kalman filter). However, there are two reasons why this may not be the preferred implementation.

1. The central node needs to handle matrix operations that increase in size as the number of sensors increases. We may want the sensors to shoulder some of the computational burden.

2. The sensors may not be able to transmit at every time step. Hence we may want to transmit after some local processing, rather than transmit raw measurements.

Having a fusion center that does not need to handle a lot of computations allows for a cheaper fusion center that is also easier to replace. The fact that

sensors may not be able to transmit at every step corresponds to message loss and other disturbances which is a ever present problem when dealing with wireless communication. Obtaining a process that is stable and accurate under these condition is greatly desirable.

We will initially assume that the sensors can transmit at every time step and concentrate on reducing the computational burden at the central node.

Transmitting Local Estimates

Our first guess would be to generate a local estimate at each sensor that extracts all the relevant information out of the local measurements and then to combine the estimates using methods outlined above. However, in general, it is not possible to use the above method. Consider K sensors being present with the k-th sensor generating a measurement of the form

$$y_{n,k} = C_k x_n + v_{n,k}.$$

and with the time steps related trough

$$x_{n+1} = Ax_n + w_n$$

Suppose we denote by Y_k the set of all the measurements from the sensor k that can be used to estimate the state x_n, i.e., the set $\{y_{0,k}, y_{1,k}, \ldots, y_{n,k}\}$. We wish to see if the local estimates formed by the sets Y_k's can be combined to yield the optimal global estimate of x_n. We can think of two ways of doing this:

1. We see that the set Y_i is linearly related to $x(k)$ through an equation of the form

$$\begin{bmatrix} y_{n,k} \\ y_{n-1,k} \\ \vdots \\ y_{0,k} \end{bmatrix} = \begin{bmatrix} C_k \\ C_k A^{-1} \\ \vdots \end{bmatrix} x_n + \begin{bmatrix} v_{n,k} \\ v_{n-1,k} - CA^{-1}w_{n-1} \\ \vdots \end{bmatrix}.$$

   However we notice that the process noise w appears in the noise vector. Thus even though the measurement noises $v_{n,k}$'s may be independent, the noise entering the sets Y_k become correlated and hence the estimates cannot be directly combined. Of course, if the process noise is absent, the estimates can be combined in this fashion (see, e.g, (Willsky et al., 1982) where the optimality in this special case was established. For a general discussion about the effects introduced by the process noise see, e.g. (Bar-Shalom, 1981; Bar-Shalom and Campo, 1986; Roecker and McGillem, 1988; Chang et al., 1997; Mori et al., 2002)).

2. We see that x_n can be estimated once the variables x_0, w_0, $\ldots$, w_{n-1} are estimated. Now Y_k is linearly related to these variables through

$$\begin{bmatrix} y_{n,k} \\ y_{n-1,k} \\ \vdots \\ y_{0,k} \end{bmatrix} = \begin{bmatrix} C_kA^n & C_kA^{n-1} & \cdots & C \\ C_kA^{n-1} & \cdots & C & 0 \\ \vdots & & & \end{bmatrix} \begin{bmatrix} x_0 \\ w_0 \\ \vdots \\ w_{n-1} \end{bmatrix} + \begin{bmatrix} v_{n,k} \\ v_{n-1,k} \\ \vdots \\ v_{0,k} \end{bmatrix}.$$

   Now the measurement noises for different sensors are uncorrelated and the estimates can be combined. However, the vector being transmitted from either of the sensors is increasing in dimension as the time step n increases. Moreover the computation required is increasing since a matrix of size growing with time needs to be inverted at every time step. Hence this is not a practical solution.

Thus we see that it is not straight-forward to combine local estimates to obtain the global estimate. We can ask the question if it is possible at all to obtain the global estimate from the local estimates. Thus imagine that the local estimates $\hat{x}_{n,k}$ were being combined in the optimal fashion. Is it possible to generate the global estimate $\hat{x}_n$? As noted above, for the special case when there is no process noise, this is indeed true. However, in general, it is not possible.

Proposition 8.2.4. *(From (Chong, 1979)) Suppose two sets of measurements Y_1 and Y_2 are used to obtain local estimates $\hat{x}_1$ and $\hat{x}_2$. Let*

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = L \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \triangleq LY.$$

Then the global estimate $\hat{x}$ can be obtained from the local estimates $\hat{x}_1$ and $\hat{x}_2$ if and only if

$$R_{YY}L^T \left(LR_{YY}L^T\right)^{-1} LR_{YX} = R_{YX}.$$

Proof. The global estimate generated from the measurements is with the help of proposition 8.1.2 given by

$$\hat{x} = R_{XY}R_{YY}^{-1}Y.$$

If it is generated from the local estimates, it is given by

$$\hat{x} = R_{X,LY}R_{LY,LY}^{-1}LY,$$

$$\Downarrow$$

$$\hat{x} = R_{XY}L^T \left(LR_{YY}L^T\right)^{-1} LY.$$

The result is thus obvious. □

If L is invertible, the condition is satisfied and hence the global estimate can be generated from the local estimates. In general, however, L would be a fat matrix and hence the condition will not be satisfied. We thus have two options:

1. Find the best possible global estimator from the space spanned by the local estimates. This is left as an exercise.

2. Find the extra data that should be transmitted that will lead to the calculation of the global estimate. We will now describe some such schemes. For these and more such strategies see, e.g., (Willsky et al., 1982),(Mori et al., 2002)–(Chong et al., 2000).

Distributed Kalman Filtering

For this section we will assume that the sensors are able to transmit information to the central node at every time step. We will use the following *information form* of the Kalman filter update equations.

Proposition 8.2.5. *Consider a random variable evolving in time as*

$$x_{n+1} = Ax_n + w_n.$$

Suppose it is observed through measurements of the form

$$y_n = Cx_n + v_n.$$

Then the measurement updates of the Kalman filter can be given by this alternate information form.

$$\begin{aligned} P_{n|n}^{-1}\hat{x}_{n|n} &= P_{n|n-1}^{-1}\hat{x}_{n|n-1} + C^T R^{-1} y_n \\ P_{n|n}^{-1} &= P_{n|n-1}^{-1} + C^T R^{-1} C. \end{aligned}$$

Proof. The equations were derived in section 8.2.1. □

The basic result about the requirements from the individual sensors can be derived using the above result.

Proposition 8.2.6. *The global error covariance matrix and the estimate are given in terms of the local covariances and estimates by*

$$\begin{aligned} P_{n|n}^{-1} &= P_{n|n-1}^{-1} + \sum_{k=1}^{K} \left(P_{n,k|n}^{-1} - P_{n,k|n-1}^{-1} \right) \\ P_{n|n}^{-1}\hat{x}_{n|n} &= P_{n|n-1}^{-1}\hat{x}_{n|n-1} + \sum_{k=1}^{K} \left(P_{n,k|n}^{-1}\hat{x}_{n,k|n} - P_{n,k|n-1}^{-1}\hat{x}_{n,k|n-1} \right). \end{aligned}$$

Proof. Proof follows by noting that the global estimate is given by

$$\begin{aligned} P_{n|n}^{-1}\hat{x}_{n|n} &= P_{n|n-1}^{-1}\hat{x}_{n|n-1} + C^T R^{-1} y_n \\ P_{n|n}^{-1} &= P_{n|n-1}^{-1} + C^T R^{-1} C. \end{aligned}$$

Since R is block diagonal, the terms $C^T R^{-1} y_n$ and $C^T R^{-1} C$ are decomposed into the sums

$$\begin{aligned} C^T R^{-1} y_n &= \sum_{k=1}^{K} C_k^T R_k^{-1} y_{n,k} \\ C^T R^{-1} C &= \sum_{k=1}^{K} C_k^T R_k^{-1} C_k. \end{aligned}$$

Noting the for the k-th sensor, the estimate and the error covariance are given by

$$\begin{aligned} P_{n,k|n}^{-1}\hat{x}_{n,k|n} &= P_{n,k|n-1}^{-1}\hat{x}_{n,k|n-1} + C_k^T R_k^{-1} y_{n,k} \\ P_{n|n}^{-1} &= P_{n|n-1}^{-1} + C_k^T R_k^{-1} C_k, \end{aligned}$$

the result follows immediately by rearranging the expressions. □

Based on this result we now give two architectures for dynamic sensor fusion.

1. In the first, rather obvious, architecture, the individual sensors transmit the local estimates $\hat{x}_{n,k|n}$. The global fusion center combines the estimates using the theorem given above. Note that the terms $\hat{x}_{n|n-1}$ and $\hat{x}_{n,k|n-1}$ can be calculated by the fusion node by using the time update equation

   $$\hat{x}_{n|n-1} = A\hat{x}_{n-1|n-1}.$$

   Similarly all the covariances can also be calculated without any data from the sensor nodes. This method is simple, especially at the sensor level. However, the fusion node has to do a lot of computation.

2. This method makes the computation at the fusion node simple at the expense of more data transmitted from the sensor node. The essential point is the observation as developed, e.g., in (Gupta et al., 2007; Gupta et al., 2009) that the term $P_{n|n-1}^{-1}\hat{x}_{n|n-1}$ can be written in terms of contributions from individual sensors, i.e.,

   $$P_{n|n-1}^{-1}\hat{x}_{n|n-1} = \sum_{k=1}^{K} z_{n,k}. \tag{8.2}$$

This can be proved using straight-forward algebraic manipulation as follows.

$$\begin{aligned} P^{-1}_{n|n-1}\hat{x}_{n|n-1} =& P^{-1}_{n|n-1}A\hat{x}_{n-1|n-1} \\ =& P^{-1}_{n|n-1}AP_{n-1|n-1}P^{-1}_{n-1|n-1}\hat{x}_{n-1|n-1} \\ =& P^{-1}_{n|n-1}AP_{n-1|n-1}\left(P^{-1}_{n-1|n-2}\hat{x}_{n-1|n-2}\right. \\ &\left. + \sum_{k=1}^{K}\left(P^{-1}_{n-1,k|n-1}\hat{x}_{n-1,k|n-1} - P^{-1}_{n-1,k|n-2}\hat{x}_{n-1,k|n-2}\right)\right). \end{aligned}$$

Thus $z_i(k)$ evolves according to the relation

$$\begin{aligned} z_{n,k} = P^{-1}_{n|n-1}AP_{n-1|n-1}\,(&z_{n-1,k} \\ &+ \left(P^{-1}_{n-1,k|n-1}\hat{x}_{n-1,k|n-1} - P^{-1}_{n-1,k|n-2}\hat{x}_{n-1,k|n-2}\right)\Big), \end{aligned} \tag{8.3}$$

which depends only on the k-th sensor's data in a recursive way. The covariances do not depend on the data and can be calculated anywhere, hence each sensor transmits the quantity

$$\gamma_{n,k} = \left(P^{-1}_{n,k|n}\hat{x}_{n,k|n} - P^{-1}_{n,k|n-1}\hat{x}_{n,k|n-1}\right) + z_{n,k}, \tag{8.4}$$

and the fusion node just calculates the sum of these quantities to arrive at the result of proposition 8.2.6. Thus at expense of more data transmitted from the sensor nodes, we have made the central node very simple.

8.3 Non-ideal Networks with Star Topology

In this section, we will give some strategies or algorithms for sensors to perform distributed estimation if the communication network suffers from limited bandwidth or message loss. These limitations are highly interesting to study since they arise commonly in the world. If we limit the bandwidth required to communicate the local estimates we can greatly reduce the required energy to transmit the estimates giving us more efficient systems. It also turns out that an implementation that highly limits the required bandwidth is easily modified to avoid any centralized administration. Avoiding centralized administration makes it possible to establish a robust ad-hoc network that may use any sensor as a fusion center that also allows for seamless modification of the number of sensors. The message loss which we will look at first is also interesting since communication is required even in high noise environments where message losses are highly likely.

8.3.1 Sensor Fusion in Presence of Message Loss

This research direction considers the following problem. Consider multiple sensors as above with a central fusion center. The sensors transmit data to the fusion center across an analog erasure link that drops messages stochastically. More formally, an analog erasure link accepts as input a real vector $i(n) \in \mathbf{R}^t$ for a bounded dimension t. At every time n, the output $o(n)$ is given by

$$o(n) = \begin{cases} i(n) & \text{with probability } 1-p \\ \emptyset & \text{otherwise.} \end{cases}$$

- The case when $o(n) = \emptyset$ is referred to as an erasure event. It implies that the channel drops the messages and the receiver does not receive any information apart from that an erasure event has occurred.

- This model assumes that the erasure events occur according to a Bernoulli process with erasure probability $1-p$. Other models, in which such events occur according to a Markov chain or other more general processes, can be considered.

- If the transmitter also knows that an erasure event has occurred, then we say that the receiver transmits an acknowledgement to the transmitter. Such an acknowledgement may always be available, may itself be transmitted across an erasure channel so that it is stochastically available, or may not be available at all.

The basic effect of the sensors transmitting across such channels is that information from the sensors is not available at the fusion center at every time step. This fact also requires some care in how the performance of the estimator is defined. Consider a realization of the erasure process such that at time n, the last transmission from sensor k was received at the fusion center at time n_k. Obviously, there is no algorithm that can provide a better estimate than the MMSE estimate of $x(n)$ given measurements $\{y_{0,1}, \cdots, y_{n_1,1}\}$, $\{y_{0,2}, \cdots, y_{n_2,2}\}$, $\cdots$, $\{y_{0,K}, \cdots, y_{n_K,K}\}$ (where we assume K sensors are present). Denote the error covariance of this estimator by P_n^{opt}. Due to the stochastic erasure process, it may be more convenient to consider the expected value of this covariance $E[P_n^{\text{opt}}]$ where the expectation is taken with respect to the erasure processes. Several questions arise:

1. What information should the sensors transmit to enable the fusion center to achieve the covariance P_n^{opt} at every time step?

2. If this covariance is not achievable, what is the best covariance that any algorithm can achieve?

3. Clearly, the error covariance at the fusion center degrades as the erasure probabilities increase. What are the conditions on the erasure probabilities so that any algorithm can achieve stability of the estimate error covariance, i.e., ensure that the expected error covariance remains bounded as $n \to \infty$?

It should be made clear that an algorithm may lead to stability of the error covariance without being optimal in the sense of achieving the covariance P_n^{opt}. In other words, the requirement in the third question is a weaker statement than the first two.

These questions are still fresh in research and only the third question has been answered (Gupta et al., 2009) generally. We will thus present the requirements needed for a bounded error and present a algorithm that achieves this, the theory will thus be useful for both designing and implementing systems. Recent developments and some special cases for questions one and two will then be described.

We now present the result below for the case when two sensors transmitting data to the fusion center across individual analog erasure links with Bernoulli erasures with erasure probabilities $1 - p_k$, $k = 1, 2$. Various other generalizations are available in the cited reference.

Theorem 8.3.1 (From (Gupta et al., 2009))**.** *Consider a process evolving as*

$$x_{n+1} = Ax_n + w_n$$

being observed using two sensors that generate measurements of the form

$$y_{n,k} = C_k x_n + v_{n,k}, \qquad k = 1, 2\,,$$

where w_n and $v_{n,k}$ are white zero mean independent noises. Let the sensors transmit information through a real vector with bounded dimension to a fusion center across analog erasure channels with Bernoulli erasures with erasure probabilities p_1 and p_2 respectively. Denote by $\rho(A_k)$ the spectral radius of the unobservable part of matrix A when the pair (A, C_k) is written in the observer canonical form and by $\rho(A_{obs})$ the spectral radius of matrix A_{obs} that corresponds to the modes observable by both sensors. Assume that the pair $(A, [C_1^T, C_2^T]^T)$ is observable.

1. *Irrespective of the information transmitted by the sensors, and the algorithm used by the fusion center, the quantity $E[P_n^{\text{opt}}]$ is not bounded as $n \to \infty$ if at least one of the following inequalities is not satisfied:*

$$p_1 \rho(A_2)^2 < 1 \tag{8.5}$$

$$p_2 \rho(A_1)^2 < 1 \tag{8.6}$$

$$p_1 p_2 \rho(A_{obs})^2 < 1. \tag{8.7}$$

2. *Conversely, if the inequalities* (8.5)–(8.7) *are satisfied, then there is an algorithm such that the corresponding expected error covariance at the fusion center is bounded as time increases.*

The full proof of this theorem is out of the scope of this text. It is important to note that the necessity of the inequalities (8.5)–(8.7) holds irrespective of the availability of acknowledgements at the sensors. The necessity part of the result follows from system theoretic considerations. The sufficiency part of the result is proved by constructing an algorithm that guarantees stability of the estimator error covariance, even though the error covariance is not P_n^{opt}. Perhaps somewhat surprisingly, the algorithm is based on the sensors transmitting local estimates of the process state based on their own measurements. The algorithm is illustrated in figure 8.6 and works as follows:

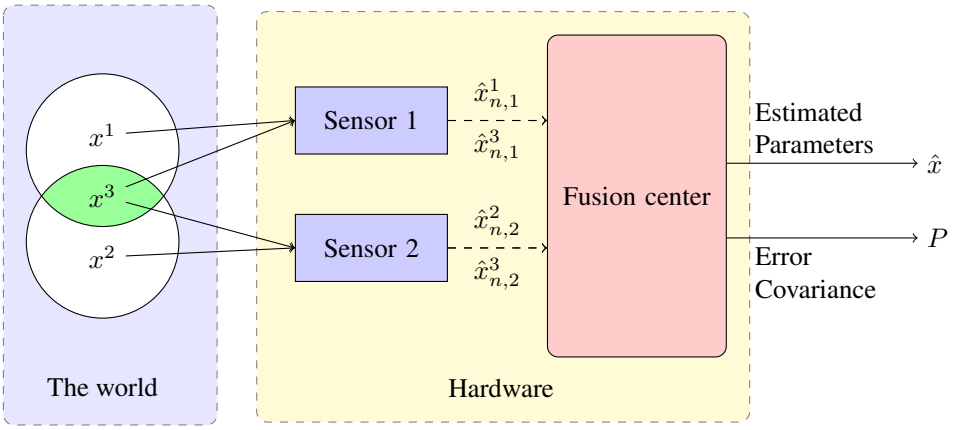


Figure 8.6 Illustration of the whole process of estimation in the presence of erasure links.

The vector being estimates consists of three parts x^1, x^2 and x^3 where the numbers are indexes and not exponents. The parts are observable from sensor 1, sensor 2 and both sensors respectively. The sensors will form and transmit local estimates in the following way

	x^1	x^2	x^3
Sensor 1	$\hat{x}^1_{n,1}$	$\emptyset$	$\hat{x}^3_{n,1}$
Sensor 2	$\emptyset$	$\hat{x}^2_{n,2}$	$\hat{x}^3_{n,2}$

where $\hat{x}^i_{n,k}$ should be understood as the estimate of the i-th part of x, at sensor k and time n. The fusion center will now form the global estimate in the following way

$$\hat{x}_n^1 = \begin{cases} \hat{x}_{n,1}^1 & \text{transmission successful from sensor 1} \\ A_2\hat{x}_{n-1}^1 & \text{otherwise} \end{cases}$$

$$\hat{x}_n^2 = \begin{cases} \hat{x}_{n,2}^2 & \text{transmission successful from sensor 2} \\ A_1\hat{x}_{n-1}^2 & \text{otherwise} \end{cases}$$

$$\hat{x}_n^3 = \begin{cases} \hat{x}_{n,1}^3 & \text{transmission successful from sensor 1} \\ \hat{x}_{n,2}^3 & \text{transmission successful from sensor 2 but not from sensor 1} \\ A_{obs}\hat{x}_{n-1}^3 & \text{otherwise.} \end{cases}$$

The estimate of the state x_n can be formed from the three components $\hat{x}_n^k$. Given this algorithm, the sufficiency of the inequalities (8.5)–(8.7) for stability of the expected error covariance can then be proved.

Given Proposition 8.2.4, it is not surprising that this algorithm cannot lead to the calculation of the optimal global estimate at the fusion center, this can also be realised from the combination rule of x_n^3 that discards the second sensors data if it has already received data. The algorithm focuses on having updated measurements and not the most accurate estimates. In fact, the optimal information processing algorithm at the sensors remains unknown in most cases. Something that is worth noting for the stability theorem is that it is quite easy to decrease the the erasure probabilities by sacrificing bandwidth if the stability constraints are not achieved. Simply sending the data twice will give an erasure probability of p^2 instead of p.

For the problem of finding the optimal stable estimate most recent advances (e.g. (Gupta et al., 2007)–(Gupta and Martins, 2009)) build from the basic algorithm identified in equations (8.2)–(8.4). Thus the sensors transmit the quantity $\gamma_{n,k}$ at every time step and the fusion center sums these quantities to generate the estimate $\hat{x}_n$. If there are no erasures, this estimate is indeed the global estimate with the optimal error covariance P_n^{opt}. However, if there are erasures, then the calculation of $\gamma_{n,k}$ requires some global knowledge. In particular, the quantity $P_{n-1|n-1}$ in (8.3) at each sensor requires the knowledge of the last time step at which the transmission from every sensor to the fusion center was successful. Notice that the data transmitted by other sensors is not required, merely the confirmation of successful transmission is enough.

One mechanism for such global knowledge can be acknowledgements transmitted from the fusion center. If such acknowledgements are available, then it was shown in (Gupta et al., 2009) that minor modifications of the algorithm outlined in equations (8.2)–(8.4) will generate the optimal global estimate at the fusion center. Depending on the problem scenario, such an assumption may or may not be realistic. If acknowledgements are also

transmitted across an analog erasure link, (Gupta and Martins, 2009) presented some further modifications to the algorithm that guaranteed that the estimation error covariance degraded continuously as a function of the probability of loss of acknowledgement. However, the optimal algorithm when acknowledgements are not available, or only available intermittently, is not known. Other special cases where such global knowledge is available can be if only one of the sensors transmits across an analog erasure link (Gupta et al., 2007) or if only one sensor transmits at any time (Hovareshti et al., 2007). Once again, in these cases, the optimal global estimate can be calculated. However, it remains unknown if the optimal global estimate can be calculated outside of these cases, or if it cannot be calculated, then what is the best performance that is achievable.

8.3.2 Sensor Fusion with Limited Bandwidth

We will now describe how to limit the bandwidth required to preform a distributed estimation to just a few bits. Optimally one would want to achieve the lower limit of communication with a single bit while still having a small error. We will show that it is possible to attain arbitrary precision while only sending a single bit from each processor given a large enough amount of sensors. This is possible for known noise distributions as well as completely unknown noise distributions something that makes it highly adaptable.

Static Sensor Fusion

Consider a limited bandwidth communication network, in which K sensors measure an unknown parameter $\theta \in [-U, U]$. The measurement x_k, from k-th sensor, is corrupted by noise n_k, which is assumed independent, zero mean, and with a pdf $p(u)$, namely $\Pr(n_k = u) = p(u)$.

$$x_k = \theta + n_k \quad \text{for } k = 1, 2, \ldots, K\,. \tag{8.8}$$

Depending on the distribution of the noise, and on the amount of information that it is transmitted, there can be the cases studied in the following subsections:

An ϵ-estimator with known noise pdf Here an ϵ-estimator is defined as an estimator providing estimates with MSE lower than ϵ^2. Assume the limited bandwidth forces each sensor to send just one bit messages $m_k(x_k)$ to the fusion center. The message is defined as

$$m_k(x_k) = \begin{cases} 1, & \text{if } x_k \in S_k \\ 0, & \text{if } x_k \notin S_k \end{cases}\,, \tag{8.9}$$

where S_k is a subset of $\mathbb{R}$ and is independent of the noise pdf. Let $\mathbb{R}_+$ denote the subset of $\mathbb{R}$ for all positive real number. The following example illustrates how this works for a uniform noise distribution.

Example 8.3.2. *(From (Luo, 2005b)) Suppose that the noise is uniformly distributed over the interval $[-U, U]$. Let $S_k = \mathbb{R}_+$ for all k. Suppose a linear fusion function that gives the estimator $\hat{\theta}$ as*

$$\hat{\theta} := f(m_1, \ldots, m_k) = -U + \frac{2U}{K} \sum_{k=1}^{K} m_k \,.$$

Then, the estimator is unbiased:

$$\mathbb{E}\left[\hat{\theta}\right] = -U + \frac{2U}{K} \sum_{k=1}^{K} \mathbb{E}\left[m_k\right] = -U + \frac{2U}{K} K \frac{U+\theta}{2U} = \theta \,.$$

Furthermore, since m_k's are independent,

$$\begin{aligned} \mathbb{E}\left[\hat{\theta} - \theta\right]^2 &= \frac{4U^2}{K^2}\, \mathbb{E}\left[\sum_{k=1}^{K} \left(m_k - \frac{U+\theta}{2U}\right)\right]^2 \\ &= \frac{4U^2}{K^2} \sum_{k=1}^{K} \mathbb{E}\left[m_k - \mathbb{E}\left[m_k\right]\right]^2 \leq \frac{U^2}{K} \,, \end{aligned}$$

where we used that the variance of a binary random variable is bounded above by 1/4. It indicates that, even with the binary message constraint, a total number of $K = U^2/\epsilon^2$ sensors are still sufficient to perform an ϵ-estimator for θ.

This can be generalized to all known noise distributions. If the $p(u)$ is given, we can still choose the message function as Eq.(8.9) with $S_k = \mathbb{R}_+$ for all k. Then

$$\begin{aligned} \Pr(m_k = 1) = \Pr(n_k > -\theta) = \textstyle\int_{-\theta}^{\infty} p(u)du \,, \\ \Pr(m_k = 0) = \Pr(n_k \leq -\theta) = \textstyle\int_{-\infty}^{-\theta} p(u)du \,. \end{aligned}$$

Then the expectation value for $\mathbb{E}\left[m_k\right]$ is obtained by

$$\mathbb{E}\left[m_k\right] = \int_{-\theta}^{\infty} p(u)du = 1 - F(-\theta), \quad k = 1, 2, \ldots, K \,,$$

where $F(\cdot)$ is the cumulative distribution function (cdf) of the noise. If one chooses the final fusion function for $\hat{\theta}$ as introduced in (Luo, 2005b), then

$$\hat{\theta} := f(m_1, \ldots, m_k) = -F^{-1}\left(1 - \frac{1}{K} \sum_{k=1}^{K} m_k\right) \,, \tag{8.10}$$

where F^{-1} is the inverse of F. By the strong law of large numbers, it follows

$$\begin{aligned}\lim_{K\to\infty}\hat{\theta} &= -F^{-1}\left(1-\lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^{K}m_k\right)\\ &= -F^{-1}(1-\mathbb{E}\left[m_k\right]) = -F^{-1}(F(-\theta)) = \theta\,.\end{aligned}$$

Suppose the noise pdf $p(u)$ is known and bounded over $[-U,U]$, then $\hat{\theta}$ obtained by Eq.(8.10) is an ϵ-estimate of θ implying a total number of $\mathcal{O}(1/\epsilon^2)$ sensors, by the following theorem:

Theorem 8.3.3. *(From (Luo, 2005b)) Suppose the noise pdf $p(u)$ is known and bounded from below by $\mu>0$ over $[-U,U]$. Let $K\geq 1/(4\mu^2\epsilon^2)$. Then the decentralized estimation scheme* (8.9) *and* (8.10) *produces an ϵ-estimator of θ.*

Proof. Notice that

$$\begin{aligned}|F(-\theta)-F(-\theta')| &= |1-F(-\theta)-(1-F(-\theta'))|\\ &= \left|\int_{-\theta}^{-\theta'}p(u)du\right| \geq \mu|\theta-\theta'| \qquad \forall\theta,\theta'\in[-U,U]\\ \Rightarrow |F^{-1}(v)-F^{-1}(v')| &\leq \frac{1}{\mu}|v-v'| \qquad \forall v,v'\in[0,1]\,,\end{aligned}$$

Then,

$$\begin{aligned}|\hat{\theta}-\theta| &= \left|-F^{-1}\left(1-\frac{1}{K}\sum_{k=1}^{K}m_k\right)+F^{-1}(1-\mathbb{E}\left(m_k\right))\right|\\ &\leq \frac{1}{\mu}\left|\frac{1}{K}\sum_{k=1}^{K}m_k-\mathbb{E}\left(m_k\right)\right|\\ \Rightarrow\ \mathbb{E}\left[\hat{\theta}-\theta\right]^2 &\leq \frac{1}{\mu^2}E\left[\frac{1}{K}\sum_{k=1}^{K}m_k-\mathbb{E}\left(m_k\right)\right]^2 \leq \frac{1}{4\mu^2K}\,.\end{aligned}$$

Thus, the variance of the estimator given by Eq.(8.10) is lower than ϵ^2 as long as

$$K\geq\frac{1}{4\mu^2\epsilon^2}\,.$$

which concludes the proof. □

This results verifies the result in example 8.3.2.

A universal ϵ-estimator for unknown noise pdf The ϵ-estimator introduced in Section 8.3.2 needs the explicit pdf $p(u)$ for the noise. However, sometimes for a large number of sensors, to characterize the measurement noise distribution would cost too much, or could be even impossible in a dynamic environment. To cope with these situations, a distributed estimator providing accurate estimates regardless the noise pdf under the bandwidth constraint is required. In this subsection, we summarize a universal distributed estimator for unknown noise pdf.

The idea, proposed in (Luo, 2005b), is to represent the estimates in binary form by quantizing the sensor measurements into the corresponding bit positions. Specifically, it tries to quantize 2^{-i} of the sensors' measurements into the i-th most significant bit (MSB), $\frac{1}{2}$ of the sensors quantize their measurement into the first MSB, $\frac{1}{4}$ of the sensors quantize their measurement to the second MSB etc. Then it can be shown that a slight modification of the statistics average of these message functions $(m_1 + m_2 + \cdots + m_K)/K$ is a unbiased estimator for θ, while its MSE is upper bounded by $4U^2/K$. The fact that the first MSB has the highest probability to be transmitted is quite reasonable and easy to motivate. The MSB will have the biggest impact on the value of the estimate and thus getting that bit correct will thus have the biggest impact in minimizing the error. It is however not obvious that the probability to be used should be 2^{-i} but it can be proven that this minimizes the upper bound on the estimated MSE by preforming proof 8.3.5 with an arbitrary probability and then minimizing the result. Arbitrary probabilities will however still attain unbiased estimates although with larger errors then the one chosen here.

The procedure of this distributed estimation scheme is illustrated in figure 8.7 and is described as follows (Xiao et al., 2006; Luo, 2005a):

1. Each measurement, x_k, in node k is quantized into the i-th MSB with probability 2^{-i}, being converged to a binary message. That is the sensor will select the i-th most interesting bit and transmit its value to the fusion center. This step can be described as following

$$\Pr(a = i) = \begin{cases} 2^{-i} & i = 1, 2, 3, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{8.11a}$$

$$m_k(x, a) = \begin{bmatrix} b(2U + x; a) \\ a \end{bmatrix}, \tag{8.11b}$$

   where the value of the random variable a indicates the position for the MSB, the notation $b(z; i)$ denotes the i-th MSB of a real number z and each sensor transmit both the values of the i-th MSB as well as which bit is sent. This information will be combined in equations (8.12)–(8.14) to get an estimate of θ.

2. The fusion center recursively computes the average of all received binary messages that are distinct (determined by, say, the sender's ID),

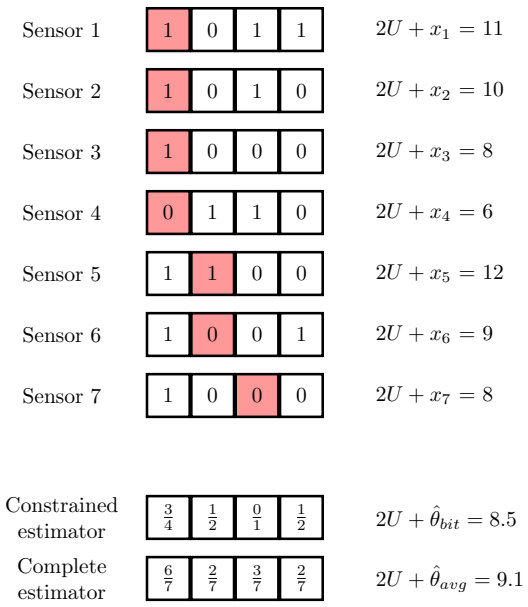


Figure 8.7 The principle used for estimation with limited bandwidth. Each sensor transmits information about one measurement made from observing $\theta = 3$ with uniform measurement noise on the interval $[-3, 3]$. The shaded bits are transmitted in the limited estimation and all bits are used in the complete estimation that calculates the mean of the measurements.

and uses it as estimator of θ.

Suppose the fusion center, which also has measurement capability, has received a total of j independent messages. Based on these messages, it can first form the sets

$$\mathcal{N}_i = \{k | a_k = i, 1 \leq k \leq j\}, \quad i = 1, 2, 3, \ldots . \tag{8.12}$$

Then, based on the received messages and its own observation x, the center can be proceed to form

$$y_i = b(2U + x; i) + \sum_{k \in \mathcal{N}_i} b(2U + x_k; a_k), \quad i = 1, 2, 3, \ldots , \tag{8.13}$$

and perform the estimate of θ

$$\hat{\theta}_j = f_j(x, m(x_1, a_1), \ldots, m(x_j, a_j)) = -2U + 4U \sum_{i=1}^{\infty} \frac{2^{-i}}{|\mathcal{N}_i| + 1} y_i \, . \tag{8.14}$$

One might think it would be more efficient to transmit $b(x; a)$ and not $b(2U + x; a)$ in order to reduce the amount of calculations required to be

done. This would however require that additional information about x would be needed at the fusion center since x can be both negative and positive. By adding $2U$ to x we guarantee that the information sent is positive and the sign of the information will thus always be the same. From this one can understand that if the sign of all measurements are known to be the same the algorithm can be simplified. This can be seen in figure 8.7, the information there could easily have been from observations on the interval [0,12] instead of [-6,6], if this would have been the case no corrections for $2U$ would be required. Figure 8.7 also gives insight into algorithms for weaker constraints of the bandwidth. It is easy to see that transmitting more bits would improve the accuracy without changing the fusion algorithm. The normal averaging estimation is simply a special case of this where every bit is transmitted. In fact we can think of the described algorithm as a combination of averaging estimation for all the different bits.

For this estimator to be useful we need it to be accurate as well and not only limit the required bandwidth. Theorems 8.3.4 and 8.3.5 show that this distributed estimator is unbiased and has an expected MSE of $4U^2/K$, where K is the number of sensors in the network:

Theorem 8.3.4. *Let $f_j(x, m(x_1, a_1), \ldots, m(x_j, a_j))$ be defined by Eq.*(8.14)*. Then for all $0 \leq j \leq K-1$*

$$\mathbb{E}\left[f_j(x, m(x_1, a_1), \ldots, m(x_j, a_j))\right] = \theta, \quad \forall \theta \in [-U, U], \quad \forall p \in \mathcal{M}_U\,, \tag{8.15}$$

where the expectation is taken with respects to the distribution of a and unknown noise, and where

$$\mathcal{M}_U = \Big\{ p(u) : \int_{-U}^{U} p(u)du = 1, \int_{-U}^{U} up(u)du = 0, p(u) \geq 0, \\ Supp(p) \subseteq [-U, U] \Big\}\,. \tag{8.16}$$

Proof. From Eq.(8.13) and (8.14), using that x_k is i.i.d to each others, we obtain

$$\begin{aligned}
&\mathbb{E}\left[f_j(x, m(x_1, a_1), \ldots, m(x_j, a_j))\right] \\
&\quad = -2U + 4U \sum_{i=1}^{\infty} \mathbb{E}\left[\frac{2^{-i}}{|\mathcal{N}_i| + 1}\left(b(2U + x; i) + \sum_{k \in \mathcal{N}_i} b(2U + x_k; a_k)\right)\right] \\
&\quad = -2U + 4U \sum_{i=1}^{\infty} 2^{-i}\, \mathbb{E}\left[b(\theta + 2U + n; i)\right] \\
&\quad = -2U + \mathbb{E}\left[\theta + 2U + n\right] = \theta\,,
\end{aligned}$$

where note that every number u in $[0, 4U]$ can be represented in binary as

$$u = 4U \sum_{i=1}^{\infty} 2^{-i} b(u; i)\,,$$

which concludes the proof. □

Theorem 8.3.5. *Let $\hat{\theta}$ be the distributed estimator of Eq.*(8.14)*. Then*

$$\mathbb{E}\,[\hat{\theta}_j - \theta]^2 \leq \frac{4U^2}{j+1}\,.$$

Proof. Similarly, from Eq.(8.13) and (8.14), using that x_k is i.i.d to each others,

$$\begin{aligned}
&\mathbb{E}\,[(\hat{\theta}_j - \theta)^2 | a_1, \ldots, a_i] \\
&\quad = 16U^2 \sum_{i=1}^{\infty} \mathrm{Var}\left[\frac{2^{-i}}{|\mathcal{N}_i|+1}\left(b(2U+x;i) + \sum_{k\in\mathcal{N}_i} b(2U+x_k;a_k)\right)\right] \\
&\quad = 16U^2 \sum_{i=1}^{\infty} 2^{-2i} \frac{\mathrm{Var}[b(2U+x;i)]}{|\mathcal{N}_i|+1} \leq 4U^2 \sum_{i=1}^{\infty} 2^{-2i} \frac{1}{|\mathcal{N}_i|+1}\,,
\end{aligned}$$

where in the last step follows from that the upper bound of $\mathrm{var}(b(2U+x;a))$ is $1/4$. Furthermore, notice that

$$\Pr(\mathcal{N}_i = r) = \left(\begin{array}{c} i \\ r \end{array}\right) 2^{-ir}(1-2^{-i})^{(j-r)}, \quad 0 \leq r \leq j\,,$$

and

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{|\mathcal{N}_i+1|}\right] &= \sum_{r=0}^{j} \frac{1}{r+1}\left(\begin{array}{c} j \\ r \end{array}\right) 2^{-jr}(1-2^{-j})^{(i-r)} \\
&= \frac{1}{j+1}\frac{1-(1-2^{-i})^{j+1}}{2^{-i}}\,.
\end{aligned}$$

Therefore, the MSE is

$$\begin{aligned}
\mathbb{E}\,[\hat{\theta}_j - \theta]^2 = \mathbb{E}\,[\,\mathbb{E}\,[(\hat{\theta}_j - \theta)^2 | a_1, \ldots, a_i]] &\leq 4U^2 \sum_{i=1}^{\infty} 2^{-2i}\,\mathbb{E}\left[\frac{1}{|\mathcal{N}_i|+1}\right] \\
&\leq \frac{4U^2}{j+1}\,,
\end{aligned}$$

which concludes the proof. □

This also proves that this estimator is an ϵ-estimator since the right hand of the inequality can be made arbitrarily small with a large enough number of sensors. It is also possible to compare this value to values for a estimator that is not bandwidth constrained. For such an estimator the fundamental lower limit on the variance is given by the Cramér-Rao lower bound (CRLB) that in our case states:

$$\mathbb{E}\,[\hat{\theta}_j - \theta]^2 \geq \frac{\sigma^2}{j+1}\,.$$

Since the variance σ^2 for a random variable on the interval $[-U, U]$ is upper bounded by U^2 and the error of the limited bandwidth estimator is as a worse case only a factor 4 from the CRLB case.

Remark 8.3.6. *The average message length is* $\sum_{i=1}^{\infty} 2^{-i}(1 + \lceil \log(i) \rceil)$ *and this sum is upper bounded by 2.5078 (Luo, 2005a).*

A 1-bit ϵ-estimator for unknown noise pdf It turns out that it is possible to limit the bandwidth even further than the average of about 2.5 bits per sensor down to a single bit per sensor. The algorithm is very similar to the previous one. The difference lies primarily in the combination of the received signals and in the fact that the sensors only transmits the i-th MSB and not its location. The algorithm is as follows:

1. The sensors all transmit the i-th MSB with the probability 2^{-i}. This can be summarized as

$$\Pr(a = i) = \begin{cases} 2^{-i} & i = 1, 2, 3, \dots \\ 0 & \text{otherwise} \end{cases}$$
$$m_k(x, a) = b(2U + x; a),$$

2. The sensors that receive the estimates m_k form the estimate

$$\begin{aligned} \hat{\theta}_j :=& f_j\left(x, m(x_1, a_1), m(x_2, a_2), ..., m(x_k, a_k)\right) \\ =& \frac{-2jU + x}{j+1} + \frac{4U}{j+1} \sum_{i=1}^{j} m(x_i, a_i)\,. \end{aligned}$$

This estimate does not make any distinction between which sensor that transmits the data which makes it very easy to update when new data arrives. The sensor doing the estimate only needs to store U, j and the sum $\sum_{i=1}^{j} m(x_i, a_i)$ in order to easily update the estimate when new data arrives. The memory requirement is thus constant for a fixed maximum number of sensors. It is also optimal in a bandwidth sense for the star topology (a more efficient communication strategy could possible be utilized in another topology with the concept of recursive doubling).

The error of this estimator is quite surprisingly no worse than that of the previous case as proven by (Luo, 2005a). One additional requirement for the error bound of

$$\mathbb{E}\left(\hat{\theta}_j - \theta\right)^2 \leq \frac{4U}{j+1}$$

is that the number of sensors is large enough that $\mathcal{N}_j$ can be approximated with $i2^{-i}$. This can simple be understood as the sensor network having enough nodes to attain the actual probability distribution. We will now prove that the estimator is unbiased.

Theorem 8.3.7. *The distributed estimator*

$$\hat{\theta}_j = \frac{-2jU + x}{j+1} + \frac{4U}{j+1} \sum_{i=1}^{j} m(x_i, a_i)$$

is unbiased.

Proof. First we look at the expected value of m_k.

$$\mathbb{E}\left(m(x_k, a_k)\right) = \mathbb{E}\left(b(2U + x_k; a_K)\right) = \mathbb{E}\left(\sum_{i=1}^{\infty} 2^{-i} b(2U + x_k; i)\right)$$

using that same reasoning as in the proof of Theorem 8.3.4 we have the following

$$\mathbb{E}\left(\sum_{i=1}^{\infty} 2^{-i} b(2U + x_k; i)\right) = \frac{2U + \theta}{4U}\,.$$

This gives that the expected value of the estimator is

$$\begin{aligned}\mathbb{E}\left(\hat{\theta}_j\right) &= \frac{-2jU + \theta}{j+1} + \frac{4U}{j+1} \sum_{i=1}^{j} \mathbb{E}\left(m(x_i, a_i)\right) \\ &= \frac{-2jU + \theta}{j+1} + \frac{4U}{j+1} \sum_{i=1}^{j} \frac{2U + \theta}{4U} = \theta\end{aligned}$$

□

That this estimator is unbiased depends solely on the probability distribution for which bit should be transmitted. If another probability is selected the expected value of m_k will not be usable.

Dynamic Sensor Fusion: Sign of innovations-KF

An alternative solution in the presence of limited bandwidth is based on the Kalman filter and the idea to send information about which direction the estimate should be corrected. This approach has the advantage of also being able to handle linear estimations. First, recall that generally distributed Kalman filter includes a prediction step and a correction step. Consider the system

$$\begin{aligned} x_n &= A_n x_{n-1} + w_n \\ y_{n,k} &= C_{n,k}^T x_n + v_{n,k}\,, \end{aligned}$$

where the driving input w_n is normally distributed with zero mean and variance Q_n and the observation noise $v_{n,k}$ is zero mean Gaussian noise and independent across sensors with noise R (Xiao et al., 2006). In this case, we

have $R = \sigma_v \mathbf{I}$. Suppose that $\hat{x}_{n-1|n-1}$ and $P_{n-1|n-1}$ are available at time n, the predicted estimate $\hat{x}_{n|n-1}$ and its corresponding covariance matrix $P_{n|n-1}$ are given by

$$\hat{x}_{n|n-1} = A_n \hat{x}_{n-1|n-1} \tag{8.18a}$$

$$P_{n|n-1} = A_n P_{n-1|n-1} A_n^T + Q_n \,. \tag{8.18b}$$

The innovation sequence

$$\tilde{y}_n := y_n - C_n^T \hat{x}_{n|n-1} \,,$$

is chosen to obtain the corrected estimate $\hat{x}_{n|n}$. To deal with the limited bandwidth, the sign of the innovation (SOI) is used to ensure that the required exchange of information among sensors is possible under one bit message constraint.

$$m(n) := \text{sign}[\tilde{\text{y}}_\text{n}] = \text{sign}[\text{y}_\text{n} - \tilde{\text{y}}_{\text{n|n-1}}] \,. \tag{8.19}$$

Due to the sign non-linearity, $p[x_n|m_{0:n-1}]$ is non-Gaussian and computation of the exact MMSE is unfeasible for most real applications. However, it is possible to decide how the estimate should be changed given the error covariances (Ribeiro et al., 2006):

$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + m_n \frac{(\sqrt{2/\pi}) P_{n|n-1} C_n}{\sqrt{C_n^T P_{n|n-1} C_n + \sigma_v^2}} \tag{8.20a}$$

$$P_{n|n} = P_{n|n-1} - \frac{(2/\pi) P_{n|n-1} C_n^T P_{n|n-1}}{C_n^T P_{n|n-1} C_n + \sigma_v^2} \,. \tag{8.20b}$$

These update equations will thus always change the estimate although always with a smaller change. Since these equations are based on simplification and approximations it is possible that the estimate becomes unstable for small C. The algorithm is two part since it requires the previous estimate $\hat{x}_{n-1}$ is order to form the SOI. For this reason the SOI-KF technique is highly suitable in an ad-hoc network where every sensor acts as a fusion center with the help of routing.

It is of interest to know how the error is affected by the SOI approach. Even at a minimal communication cost, the SOI-KF is strikingly similar to the regular KF (Xiao et al., 2006). To prove it, let us rewrite the SOI-KF correction as

$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + \frac{P_{n|n-1} C_n}{C_n^T P_{n|n-1} C_n + \sigma_v^2} \tilde{m}_{n|n-1} \,, \tag{8.21}$$

where

$$\tilde{m}_{n|n-1} := \sqrt{(2/\pi)\, \mathbb{E}\,[\tilde{y}_{n|n-1}^2]}\, m_n \,.$$

Notice that the units of $\tilde{m}_{n|n-1}$ and $\tilde{y}_{n|n-1}$ are the same, and

$$\mathbb{E}\left[\tilde{m}_{n|n-1}\right] = \mathbb{E}\left[\tilde{y}_{n|n-1}\right] = 0$$
$$\mathbb{E}\left[\tilde{m}_{n|n-1}\right]^2 = \frac{2}{\pi}\,\mathbb{E}\left[\tilde{y}_{n|n-1}\right]^2,$$

which indicates that Eq.(8.21) is identical to the KF update if replacing $\tilde{m}_{n|n-1}$ with the innovation $\tilde{y}_n$. It is not difficult to show that the MSE increases when using the SOI-KF is as much as the KF would incur when applied to a model with $\pi/2$ higher observation noise variance (Xiao et al., 2006; Ribeiro et al., 2006). The algorithm is thus highly useful.

8.4 Network with Arbitrary Topology

The results above assumed the presence of a star topology in which one central node had access to local estimates from every other node. It was essentially a two step procedure: first all the nodes transmit local estimates or local measurements to the central node and then the central node calculates and transmits the weighted sum of the local estimates back. However, what is required is a weighted average. Thus, we can generalize the approach to an arbitrary graph at the expense of more time being employed. The generalization is along the lines of average consensus algorithms that have been recently considered by many people (see, e.g., (Olfati-Saber and Murray, 2004)–(Jadbabaie et al., 2003)). An example of arbitrary topology networks is illustrated in Fig 8.8. For now, we will only cover the basics for static sensor fusion. There exists deeper and more refined theories in consensus algorithms that are easily applicable to allow for better performance in respect to convergence speed and stability when erasure links are applied. There also exists theory on how to chose the communication network to improve convergence since it turns out that simply adding more connections can in fact worsen the convergence.

8.4.1 Static Sensor Fusion with Limited Communication Range

Due to the limited communication range, some of the sensors can not send message to the fusion center. In such cases, we can treat the networks as the static sensor fusion for arbitrary graphs without a fusion center. We can still achieve good estimates and each extra sensor that belongs to the same connected graph will improve the accuracy of the estimate.

Consider K nodes each with access to a scalar value being connected according to an arbitrary (but time-invariant) connected graph. Suppose we want each node to calculate the average of all the local estimates. One way

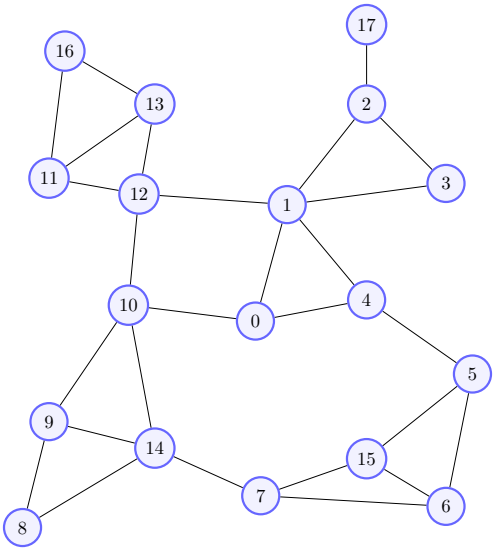


Figure 8.8 An example of arbitrary topology network with nodes and links (solid lines indicating that there is message communication between nodes). In this network, there is no node acting as fusion center.

to do that is if each node implements the dynamical system

$$x_{n+1,k} = x_{n,k} + h \sum_{j \in \mathcal{N}_i} (x_{n,j} - x_{n,k}) \ ,$$

where $x_{n,k}$ is the value for state x_k at time n, and h is a small positive constant. On stacking the states of all the nodes, the entire system evolves as

$$X_{n+1} = (I - hL)X_n,$$

where $X_n = [x_{n,1}, \ldots, x_{n,K}]^T$, and L is the *Graph Laplacian* matrix. If the underlying graph is connected, L has the following properties:

1. It is a symmetric positive-definite matrix. Thus the dynamics is stable (assuming h is small enough) and reaches a steady-state.

2. Each row sum is 0. Thus any vector with identical components is an equilibrium and corresponds to a eigenvalue of 0.

3. Each column sum is 0. Thus the sum of entries X_n is conserved at every time step.

Because of these three properties together with the fact that there exist no other equilibrium than that given by fact 2 (Olfati-Saber et al., 2007), it is easy to see that each entry must converge to the average of the sum

of the initial conditions. This algorithm can then be readily extended for calculating *weighted* averages of *vectors* (Spanos et al., 2006; Xiao et al., 2005). If the initial values are given by the vectors $x_{0,k}$, each node calculates the following:

$$x_{n+1,k} = x_{n,k} + hW_k^{-1} \sum_{j \in \mathcal{N}_i} (x_{n,j} - x_{n,k}) \ ,$$

where $\mathcal{N}_i$ denotes the set of sensors connected to i-th sensor. In our case, we let $x_{0,k}$ to be the local estimate values and W_k^{-1} to be inverse of the local estimation error covariance, and obtain the required weighted average. Since both the weighted and standard version converges to the same estimate as forming the corresponding averages with global information the error for the averages will be the same as in the global case and standard techniques for calculating the variance can be applied.

8.5 Computational Complexity and Communication Cost

The efficiency of implementation of estimation algorithms can be characterized in terms of computational complexity and communication cost. Conventionally, the computational complexity of an algorithm is measured by the amount of basic operations such as float-point arithmetic performed. The computational complexity is commonly expressed by using the $\mathcal{O}$ notation, which describes the limiting behaviour of a function. The $\mathcal{O}$ notation suppresses the multiplicative constants and lower order terms. For example, if the time running requirement for an algorithm is at most $5n^3 + 100n^2$, then we say that the computational complexity is $\mathcal{O}(n^3)$. In the multivariate case it is important to keep the highest order term from all the involved parameters in order to get a usefull representation of the system. On the other hand, communication cost of an algorithm refers to the communication resources required in terms of amount and size of the exchanged messages in bytes or bits. We express the communication cost by using the $\mathcal{O}$ notation as well.

It is important to analyze the computational complexity and communication cost for distributed estimation algorithms, especially when one designs algorithms for a sensor network. In these networks, larger computational complexity requirement and communication cost always entail the high risk of slower response speed, smaller transmit rate and thus poorer performance in practice, though the theoretical performance for the algorithm might be much better. In fact, due to the limited computational capability of sensors, we sometimes have to redesign A_{alg}, or implement an approximate algo-

rithm $\tilde{A}_{\text{alg}}$ which has lesser computational complexity but may still provide acceptable performance.

8.5.1 On Computational Complexity

Before deploying an algorithm for sensors networks, it is desirable to check whether the sensor nodes have enough computational capability to perform the computation as designed. Suppose that the sensors are designed to produce their estimates using some algorithm A_{alg}. We analyse A_{alg}'s worst-case computational cost requirements as a function of the size of its input (in terms of the $\mathcal{O}$-notation). Here we assume that arithmetic or basic operation with individual elements has complexity $\mathcal{O}(1)$. Thus, the computational complexity of a matrix addition, multiplication, and inversion are $\mathcal{O}(m^2)$, $\mathcal{O}(m^3)$ and $\mathcal{O}(m^3)$ respectively, where $m \times m$ is the size of the matrix.

Example 8.5.1. *Consider the combining estimator of Section 8.2.1. According to Proposition 8.2.1, some matrix inversions and multiplications to find the MMSE estimate of X are needed. Denote the largest size of the vector Y and X is M. Then the computational complexity of the estimators is $\mathcal{O}(M^3)$.*

Example 8.5.2. *Consider the static sensor fusion of Section 8.2.1. According to Proposition 8.2.2, similarly to Example 8.5.1, we need to perform the matrix inversions and multiplications. Notice that in this case, we need K times matrix inversions and multiplications in each iteration. Thus the computational complexity of the static sensor fusion is $\mathcal{O}(KM^3)$.*

Example 8.5.3. *Consider the local estimator of Section 8.2.2. According to the method, we need to perform the matrix inversions and multiplications. Notice that in this case, the size of the matrix is not M, but nM, where n is the time step. Thus the computational complexity of the estimator is $\mathcal{O}(n^3M^3)$.*

Example 8.5.4. *Consider the Kalman Filtering of Section 8.2.2. According to the method, we need to perform the matrix inversion and multiplications. Similar to Example 8.5.2, we need K times matrix inversions and multiplications per iteration. Thus computational complexity of distributed Kalman filtering is $\mathcal{O}(KM^3)$.*

8.5.2 On Communication Cost

Due to the limited communication resources for the network, before implementing a distributed estimator, we need to analyze the communication cost as well. We define the number of messages exchanged by the sensors as the communication cost.

Table 8.1 Summary of the time complexity and communication cost. In the table, M is the at most size of the tracked signal, n is the time step, while K is the number of the sensors in the network. R represents the extra communication cost for the routing.

Topology	Signal	Algorithm	Complexity	Cost
Star	$\mathbb{R}^M$	Different Estimators in Section 8.2.1	$\mathcal{O}(M^3)$	$\mathcal{O}(K)+R$
	$\mathbb{R}^M$	Static Sensor Fusion in Section 8.2.1	$\mathcal{O}(KM^3)$	$\mathcal{O}(K)+R$
	$\mathbb{R}^M$	Transmission Local in Section 8.2.2	$\mathcal{O}\left((n+M)^3\right)$	$\mathcal{O}(K)+R$
	$\mathbb{R}^M$	Distributed Kalman Filter in Section 8.2.2	$\mathcal{O}(KM^3)$	$\mathcal{O}(K)+R$
Arbitrary	$\mathbb{R}^1$	Static Sensor Fusion in Section 8.4.1	$\mathcal{O}(\mathcal{N}_k^3)$	$\mathcal{O}(K)$

Example 8.5.5. *In the network with star topology mentioned in the Section 8.2.2, every sensor need sharing sending messages to the center fusion. Thus the total communication cost is $\mathcal{O}(K)$ for each iteration, where K is the number of sensors in the network. If the star topology relies on routing we also have a maximum routing length R that is added to the communication cost.*

Example 8.5.6. *In the network with arbitrary topology mentioned in the Section 8.4.1, every sensor needs sending its messages with its neighbors. Since via wireless communication channels, sensor can broadcast its messages to all sensors inside its communication range, the total communication cost is $\mathcal{O}(K)$ per iteration.*

8.5.3 Summary of the computational complexity and communication cost

In this section, we have studied the computational complexities and communication cost for the distributed estimation methods summarized in the preceding sections. Now, we summarize the result in Table 8.1.

In Table 8.1, **Signal** represents the signal tracked by the sensors network, **Complexity** represents the computational complexity, whereas **Cost** represents the communication cost. It is worth mentioning that the R in the **Cost** represents the extra communication cost used for routing when the sensors only have limited communication range.

The table shows that without considering routing, the communication costs are approximately same for the networks with star and arbitrary topology. However, the computational complexities vary for different algorithms.

Generally, transmitting local estimation (in Section 8.2.2) needs most running time compared to other algorithms used in star topology. Moreover, dynamic sensor fusion (in Section 8.2.2) needs more running time than the static sensor fusion. Considering both the computational and communication complexities are important when designing systems in order to limit the impact of any bottlenecks. If the available hardware have good processors but bad sensors with low bandwidth going with heavy computational algorithms is desirable.

8.6 Conclusion

This chapter introduced basic notions of distributed estimation theory, and some implications for the applications with and without considering the limitations in the networks. Moreover, an analysis of the computational complexity and communication cost of these distributed algorithms was performed. Generally, the less the limited capability of the network and the greater the knowledge of the physical phenomenon, the lower the complexity of the resulting estimators. Nevertheless, it is often possible to establish accurate distributed estimators in the networks. We remark that, this study we gave here is an essential overview on some key aspects of distributed estimation. Much more can be summarized (e.g., the convergence or consensus properties of the distributed estimators).

Problems

Problem 8.1

Given a vector of random variables Y that is related to another vector of random variables X, describe briefly what is the best linear estimator of X if one observes an outcome of Y.

Problem 8.2 Mean square (MS) estimation (Ex.5.21 in (**?**))

Let X be a real-valued RV with a pdf of $f_X(x)$. Find an estimate $\hat{x}$ such that the mean square error of x by $\hat{x}$ is minimized when no observation is available.

Problem 8.3 ML estimates of mean and variance of Gaussian random variables

Consider n independent random samples from a Gaussian distribution $N(\mu, \sigma^2)$. Let $\theta = (\mu, \sigma)$, that is $\theta_1 = \mu$ and $\theta_2 = \sigma$. Find the Maximum-Likelihood (ML) estimates of μ and σ.

Problem 8.4 MMSE estimator

In many situations, one has to estimate x from some noisy measurements y that are a linear function of x plus some noise. Let X be a vector of random variables having zero mean. Suppose that Y is a vector of random variables related to X such that if x is an outcome of X, then an outcome of Y is $y = Hx + v$, where H is a constant matrix and v is a zero mean Gaussian noise having covariance R_v, with v independent of X. Then, the MMSE estimate of X is given that $Y = y$ is

$$P^{-1}\hat{x} = H^T R_v^{-1} y$$

with error covariance

$$P^{-1} = R_X^{-1} + H^T R_v^{-1} H\,.$$

Now, consider a network of n sensors. Let X be a random variables observed by each sensor by the noisy measurement $y_i = H_i x + v_i$ and $i = 1, \ldots, n$, where all the noises are uncorrelated with each other and with X. Let the estimate based on all the measurement be $\hat{x}$ and let $\hat{x}_i$ the estimate based on only the measurement y_i. Then,

$$P^{-1}\hat{x} = \sum_{i=1}^{n} P_i^{-1}\hat{x}_i$$

where P is the estimate error covariance corresponding to $\hat{x}$ and P_i is the estimate error covariance corresponding to $\hat{x}_i$, with

$$P^{-1} = \sum_{i=1}^{n} P_i^{-1} - (n-1)R_X^{-1}\,.$$

The above estimators, by the assumption that H_i is the i-th row of the matrix H, give the same estimate. Assume that R_X and R_v are diagonal matrixes. Motivate wether the first estimator requires more computations than the second estimator and suggest which one is best for a sensor network.

Problem 8.5 Distributed MMSE estimator

We would like to estimate a vector of unknown constant parameters $x \in \mathbb{R}^{\mathrm{m}}$ using a network of n distributed sensors. Each sensor makes a noisy measurement

$$y_i = H_i x + v_i \qquad i = 1, \ldots, n.$$

Where H_i is an known matrix relating the unknown parameter to the measurement, v_i is a Gaussian noise with zero average and covariance matrix R_{v_i}. Moreover v_i's are assumed statistically independent noises. In vector notation one can formulate $y = Hx + v$, where y, H and v are $n \times 1$ vectors of y_i, H_i and v_i. Show that the maximum likelihood (or MMSE) estimate of x given y is

$$\hat{x} = \left(\sum_{i=1}^{n} H_i^T R_{v_i}^{-1} H_i\right)^{-1} \sum_{i=1}^{n} H_i^T R_{v_i}^{-1} y_i\,.$$

Problem 8.6 Unknown mean in Gaussian noise (Example C.1 in (Gustafsson, 2012))

Consider an unknown mean in Gaussian noise,

$$y_k = x + e_k, \quad e_k \in \mathcal{N}(0, \sigma^2)\,.$$

Find the mean and variance of the sample average. Show that the sample average is the minimum variance estimator.

Hint: use the CRLB.

Problem 8.7 Moments method (Example C.9 and C.10 in (Gustafsson, 2012))

The method of moments is general not efficient and thus inferior to the ML method. However, in many cases it is easier to derive and implement. For Gaussian mixtures, the MLE does not lead to analytical solutions so numerical algorithms have to applied directly to the definitions, where whole data vector has to be used. Using the method of moments, closed expressions can be derived as functions of reduced data statics.

The key idea is to estimate the first p moments of data, and match these to the analytical moments of the parametric distribution $p(y|x)$:

$$\begin{aligned}
\mu_i &= E[y_k^i] = g_i(x)\,, \quad i = 1, 2, \ldots, p \\
\hat{\mu}_i &= \frac{1}{N}\sum_{k=1}^{N} y_k^i\,, \quad i = 1, 2, \ldots, p \\
\mu &= g(x)\,, \\
\hat{x} &= g^{-1}(\hat{\mu})\,.
\end{aligned}$$

Now consider a Gaussian mixture

$$p(y|x) = \alpha\mathcal{N}(y; 0, \sigma_1^2) + (1-\alpha)\mathcal{N}(y; 0, \sigma_2^2),$$

where

$$x = (\alpha, \sigma_1^2, \sigma_2^2)^T\,.$$

Assume that those σ are known. Using the method of moments, find the estimation of α. If the variances are unknown, find the estimation of x.

Problem 8.8 Estimating reflectance

We want to measure the reflectance of a surface, for this task we are given three lasers and three sensors that are suitable for the different lasers wavelength. The lasers operate at the wavelengths 416nm, 543nm and 611nm with output powers at 0.05W, 0.1W and 0.075W respectably. The sensors has measurement variance of 0.005W^2, 0.02W^2 and 0.01W^2.

The measured values are 0.0459 W, 0.0809 W and 0.0682 W.

(a) Assume all the measurements are sent to a fusion center. Express the relation between the reflectance and the measured values.

(b) Assume you have no information about the value of the reflectance. Calculate what each sensor should transmit to the fusion center and form the estimate of the reflectance and the error using this information.

(c) Repeat the calculations done in (b) but now include the fact that the reflectance must be a value between 0 and 1. Did the error change? Explain any changes.

Problem 8.9 Error for 1-bit estimator

Prove that $\mathbb{E}\,(\hat{\theta}_j - \theta)^2$ is upper bounded by $\dfrac{4U}{j+1}$ when $\hat{\theta}_j$ is the 1-bit estimator given by

$$\hat{\theta_j} = \frac{-2jU + x}{j+1} + \frac{4U}{j+1} \sum_{i=1}^{j} m(x_i, a_i)\,.$$

Problem 8.10 Distributed detection/estimation

A set of N nodes is randomly deployed on a field. Every node makes observations on an unknown parameter $\theta \in [-1, 1]$. The observations are corrupted by an additive noise

$$x_k = \theta + v_k, \qquad k = 1, 2, \ldots, N\,,$$

where v_k is the noise, which is modeled as a random variable. These random variables are assumed to be independent and identically distributed and with zero mean. In particular, they are uniformly distributed over $[-1, 1]$, with a probability distribution function (pdf)

$$p(v) = \frac{1}{2}, \qquad \text{if } v \in [-1, 1]\,.$$

To get an accurate estimate of the parameter θ, each node reports its observations to a fusion centre. However, due to message losses and medium access control protocol, each node is allowed to transmit a message composed only by one bit. In other words, each node reports a message $m_k(x_k) \in \{0, 1\}$ to the fusion center. The bit of the message is chosen as

$$m_k(x_k) = \begin{cases} 1\,, & \text{if } x_k \geq 0 \\ 0\,, & \text{if } x_k < 0\,. \end{cases}$$

(a) Find the expectation $\mathbf{E}(m_k)$ and variance of the one-bit message $\mathbf{E}(m_k - \mathbf{E}(m_k))^2$ for node k.

(b) Prove that $\mathbf{E}(m_k - \mathbf{E}(m_k))^2$ is bounded above. Find the upper bound.

(c) Suppose that the fusion center uses a final fusion function f and estimator $\hat{\theta}$ to decide upon the parameter given by

$$\hat{\theta} := f(m_1, \ldots, m_N) = \frac{2}{N} \sum_{k=1}^{N} m_k - 1\,.$$

Find $\mathbf{E}(\hat{\theta})$ and $\mathbf{E}(\hat{\theta} - \theta)^2$.

(d) Suppose we want the variance of estimate $\hat{\theta}$ less than ϵ. What is the minimum number of nodes N to deploy so that such a variance bound is satisfied?

Chapter 9

Distributed Learning

Sensor nodes offer wide possibilities for distributed learning. In this chapter we will explore different techniques of using learning algorithms in WSNs. One of the reasons for using WSNs is that we are interested in learning something about a phenomenon that they are sensing. There are many phenomena, both natural and man made, that we do not know how they work and WSNs allow us much greater freedom and autonomy than other sensing equipment. Thus robust and precise techniques for distributed learning using WSNs are desired. Towards this goal, we will in this chapter study some general techniques for learning including the supervised learning model and some time series models. We then reshape and remodel these techniques to suite WSNs, while considering both the case of star networks and the more general topologies. Special consideration is given to meet the strict demands on energy conservation ever present in WSNs and we work towards minimizing the required communication needs.

9.1 Learning in General

Learning is a very broad concept and in this chapter we are only going to consider some prominent methods among the many learning methods that are available. The type of learning we are interested in is the kind where from some data or observations, we develop some way of giving information of how the phenomena looks outside the given data. In other words, the goal is to predict something. This could for example be predicting future behavior by using data from the past or classifying a new object given data on other similar ones. It could also be predicting the value of something like the temperature in a spot given some measurements at other places and times. The possible scope is very wide, here we are going to limit us to the cases of classification and estimation in supervised learning and prediction of future behavior using ARMA-time series.

9.1.1 Supervised Learning

In supervised learning we are interested in finding a so called decision rule, i.e., a function, g that can tell us how something behaves given some input. Let X and Y be $\mathcal{X}$-valued and $\mathcal{Y}$-valued random variables, respectively. The set $\mathcal{X}$ is known as the feature, input or observation space and the set $\mathcal{Y}$ is known as the label, target or output space. These sets can in general be complicated but for our need in this chapter we will restrict us to the cases of classification and estimation problems. These correspond to binary classification, $\mathcal{Y} = \{0,\ 1\}$, and regression, $\mathcal{Y} = \mathbb{R}$, respectively. We also restrict the input space to $\mathcal{X} \subseteq \mathbb{R}^d$.

We seek to find a decision rule g that can couple $\mathcal{X}$ and $\mathcal{Y}$, or more explicitly, given a X we want to find what Y corresponds to it. To quantify how well the decision rule works we introduce a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and seek a decision rule that minimizes the expected loss,

$$\mathbb{E}\left\{l(g(X), Y)\right\}. \tag{9.1}$$

In the case of classification we will use the zero-one loss function defined as

$$l(y, y') := \begin{cases} 0 \text{ if } y = y' \\ 1 \text{ if } y \neq y' \end{cases}. \tag{9.2}$$

In the case of estimation we will use the squared error:

$$l(y, y') := \|y - y'\|^2 . \tag{9.3}$$

Example 9.1.1. *Consider the classification loss function in (9.2). In this example we will explain how this loss function can tell us how well a certain decision rule performs. Say that we are interested in predicting whether or not a person's cancer tumor is benign. We assume that we get some data about the person and the tumor to give our decision rule. Then we can let our decision rule predict if a group of persons have benign cancer or not. For each person the loss function will tell if the decision rule was right, giving us a 0, or wrong, giving us a 1. If we sum these number for all persons we arrive at a measure of how well the decision rule does that can be compared to other decision rules.*

Example 9.1.2. *Consider the classification loss function in (9.3). Now assume we want to predict the temperature at some location in a room. We have some decision rule to do this but want to know how well it performs. This can be done by measuring the temperature at different locations and letting the decision rule predict the outcome of these measurements given their position. Then the results are compared using the loss function. If the measurement said 20.4°C and the decision rule predicted* $19.7°C$ *we for example get* $l(20.4, 19.7) = \|20.4 - 19.7\|^2 = \|0.7\|^2 = 0.49$. *By summing these numbers up for all the measurements we get a number that tells how good the decision rule does and helps us compare different decision rules.*

If we were to know the joint probability distribution, $\mathbf{P}_{XY}$, we could simply minimize (9.1) for the estimation problem by choosing

$$g(x) := \mathbb{E}\{Y|X = x\},$$

and for the classification problem:

$$g(x) := \begin{cases} 1 & \text{if } \mathbf{P}\{Y = 1|\ X = x\} > \mathbf{P}\{Y = 0|\ X = x\} \\ 0 & \text{otherwise} \end{cases} .$$

We will denote the minimum loss achievable by

$$L^\star := \min_g \mathbb{E}\{l(g(X), Y)\}.$$

However, we rarely know $\mathbf{P}_{XY}$ beforehand, which makes the solutions above impractical. In supervised learning we assume no knowledge of $\mathbf{P}_{XY}$. The idea in supervised learning is instead to use a collection of training data, i.e., a set of input-output pairs that are used to find g. We denote this training data by

$$S_m := \{(x_i, y_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}\,.$$

We will assume that this data arises from some stochastic process and that $S_m = \{(x_i, y_i)\}_{i=1}^m$ are independently and identically distributed (i.i.d.) with $(X_i, Y_i) \sim \mathbf{P}_{XY}$.

We define a learning algorithm as the sequence $\{g_m\}_{m=1}^\infty$ of data dependent decision rules $g_m : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^m \to \mathcal{Y}$ where we do not assume anything regarding $\mathbf{P}_{XY}$. In each step of this learning algorithm a decision rule $g_m(\cdot, S_m)$ is constructed, this process is called training and the method used is called a training algorithm. Given this learning algorithm $\{g_m\}_{m=1}^\infty$ we get the expected loss at step m given the random training data S_m by

$$L_m = L_m(S_m) = \mathbb{E}\{l(g_m(X, S_m), Y)|\ S_m\}.$$

Ideally we would like our learning algorithm to perform just as well as if we would have know $\mathbf{P}_{XY}$ beforehand. To quantify this wish, we introduce the notion of universally consistent in the following definition:

Definition 9.1.3. A learning rule $\{g_m\}_{m=1}^\infty$ is universally consistent if and only if $\mathbb{E}\{L_m\} \to L^\star$ for all distributions $\mathbf{P}_{XY}$ with $\mathbb{E}\{Y^2\} < \infty$.

Example 9.1.4. *To understand how how a learning rule might be universally consistent we consider the k-nearest neighbor rule. Here we have the decision rule according to*

$$g_m(x, S_m) = \sum_{i=1}^m W_m^i(x, S_m) y_i$$

with

$$W_m^i(x, S_m) \begin{cases} \frac{1}{k} & \text{if } x_i \text{ is among the } k \text{ nearest neighbors to } x \\ 0 & \text{otherwise.} \end{cases}$$

This learning rule will average the output form the k nearest training samples to estimate Y. Assuming that $\mathbb{E}\{\|Y\|\} < \infty$ *this learning rule will be universally consistent if*

$$k \to \infty, \qquad k/m \to 0.$$

This means that we consider an ever larger number of nearest neighbors but an ever decreasing fraction of the total training sample number. This gives us an ever bigger number of samples to average and an ever smaller part of the input space to consider. Thus it is reasonable that this decision rule will perform well.

To continue our exploration of the supervised learning we will stat looking at a special class of learning algorithms called kernel methods.

Kernel Methods

Kernel methods are a popular class of learning algorithms developed for the supervised learning problem. It works by using so called kernel functions, i.e., positive semi-definite functions $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. By positive semi-definite function we mean that for all $n \in \mathbb{N}$ and for all $x_1, x_2, ..., x_n \in \mathcal{X}$, denoting the $n \times n$ matrix $K = (k_{ij})$ where $k_{ij} = \kappa(x_i, x_j)$, we have that K is a symmetric and positive semi-definite matrix.

These kernel functions help measure similarity between input data. The decision rule then associates each input $x \in \mathcal{X}$ to a weighted average of training data outputs depending on the inputs similarity to training data inputs. One popular decision rule is depicted below:

$$g_m(x) := g_m(x, S_m) = \begin{cases} \frac{\sum_{i=1}^m \kappa(x, x_i) y_i}{\sum_{i=1}^m \kappa(x, x_i)} & \text{if } \sum_{i=1}^m \kappa(x, x_i) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{9.4}$$

As stated now (9.4) works fine for the estimation problem but produces unavailable values of Y for the classification problem. Most certainly the obtained value will not be 0 or 1. By rounding to which ever of these is closer however we get the right results.

There are infinite possibilities in the choice of the kernel function $\kappa(\cdot, \cdot)$. How to make the specific choice for an application is not an easy task and is in its own a big field. Here we will not restrict us to some specific case but to get a feel for how they might look some commonly used kernel functions

Table 9.1 Common kernels

$\kappa(x, x')$	Name
$\left.\begin{array}{l} 1 \text{ if } \|x - x'\|^2 < r_m^2 \\ 0 \text{ otherwise} \end{array}\right\}$	Naive kernel
$x^T x'$	Linear kernel
$(1 + x^T x')^d$	Polynomial kernel
$e^{-\|x-x'\|^2/\sigma}$	Gaussian kernel
$\left.\begin{array}{l} 1 - \|x - x'\|^2 \text{ if } > 0 \\ 0 \text{ otherwise} \end{array}\right\}$	Epanechnikov kernel
$1/(1 + \|x - x'\|^{d+1})$	Cauchy kernel

can be found in Table 9.1. In the case of the naive and the Gaussian we have an unknown parameter, r_m and σ respectively. These are there to give kernel more flexibility by modifying the kernels' reach.

To show how we can use (9.4) and the kernels in Table 9.1 there follows two examples below. One handles the case of classification and the other estimation.

Example 9.1.5. *To show how we can use the (9.4) for the classification problem we consider the data set in Figure 9.1. Here the input space is the two dimensional position in the x_1x_2-plane, $x_1, x_2 \in [0, 5]$. The points marked with an X are negative results (0) and the points marked with an O are positive (1). We now seek a decision rule to classify all points in the plane. we can think of the measurements as whether or not there exists oil under the ground at a given location in an area. To be able to predict where we have oil and where it is absent we use the decision rule in (9.4) and use a modified Gaussian kernel from Table 9.1:*

$$\kappa(x, x') = e^{-\|x-x'\|^2/0.3}\,.$$

Using this kernel and the underlying data we get a decision rule that give the results in Figure 9.2. Here we have rounded the results from (9.4) to the closer of 0 or 1. The grey area shows where the decision rule predicts a positive outcome, in other words an O. The white area represents a negative outcome, i.e., an X. We can see that the decision rule classifies all our training data in the right way indicating good performance.

Example 9.1.6. *To show how we can use the (9.4) for the estimation problem we consider the data set in Figure 9.3. Here the input space is the two dimensional position in the x_1x_2-plane, $x_1, x_2 \in [-2.5, 2.5]$. The blue circles with a number in are measured points. The lines in the background are*

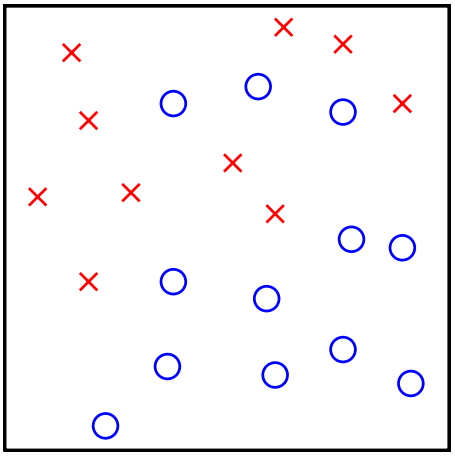


Figure 9.1 Data for Example 9.1.5.

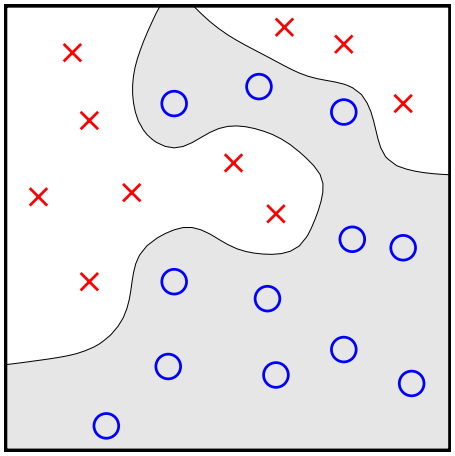


Figure 9.2 Solution for Example 9.1.5.

contour lines with marked values of the measured quantity. It follows the function $y = 4x_1^2 + 6x_2^2$. *We now seek a decision rule to estimate all points in the plane. We can think of the measurements as the height at different locations on a piece of land. To estimate we use the decision rule in (9.4) and, as in Example 9.1.5, use a modified Gaussian kernel from Table 9.1:*

$$\kappa(x, x') = e^{-\|x-x'\|^2/0.3}\,.$$

The result can be seen in Figure 9.4. The measured points are still visible but in the background we now have the decision rule contour map. It ranges from dark blue, 0-5, all the way to dark red, 35-40. As we can see the overall shapes look quite similar, although the decision rule has not got as smooth ellipse contours. The big differences can mostly be found at the places that are most distant from the training data inputs. This makes sense as we have less information at these places.

There can for example be shown that both the naive and the Gaussian kernels can enable universally consistent learning by using (9.4). This is part of the very general Stone's theorem which however is beyond the scope of this book. We merely take this as an indicator that these methods can give us very good results.

Principle of Empirical Risk Minimization

Now returning to the task at hand, finding the minimizer of (9.1). This is a bit tricky since we have no information on $\mathbf{P}_{XY}$. The principle of empirical risk minimization requires the learning rule to minimize a data dependent approximation of (9.1). If we have a big sample of training data the average

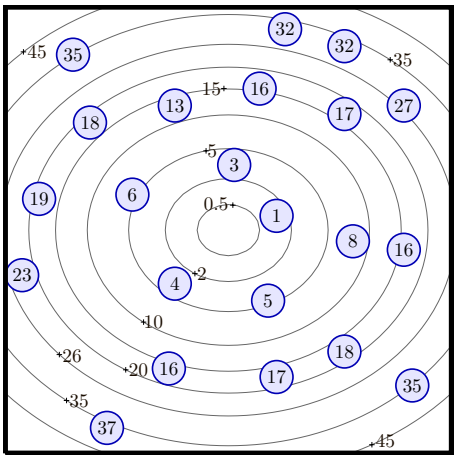


Figure 9.3 Data for Example 9.1.6.

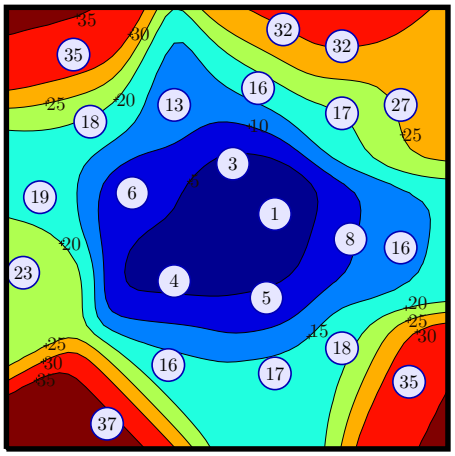


Figure 9.4 Solution for Example 9.1.6.

of the losses given the individual samples should give a good approximation of (9.1). This motivates the following equation:

$$g_m^\lambda = \arg\min_{f \in \mathcal{F}} \left[\frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i) + \lambda \|f\|_{\mathcal{F}}^2 \right]. \tag{9.5}$$

Here the first term is the empirical loss of the decision rule and the second term acts as a complexity controller and thus regularizes the optimization. $\lambda \in \mathbb{R}$ is a parameter that governs the trade off between the two. We have also restricted the optimization variable, i.e., function, to be in a Hilbert space $\mathcal{F}$ with the norm $\|\cdot\|_{\mathcal{F}}$. Now if we let $m \to \infty$ and $\lambda \to 0$ we should arrive at a good approximation of (9.1). Equation (9.5) will be used in the next section and we shall see that we can find an analytic solution to the problem for certain functions using the squared error loss function. For now let us look at Example 9.1.7.

Example 9.1.7. *To see how (9.5) can be used let us consider the data set found in Figure 9.5, we use the same notations as in Example 9.1.5. Let us say we want to find a line to split the x_1x_2-plane, $x_1, x_2 \in [0, 5]$, into two parts, telling the algorithm how to classify the points. We choose this type of function for its simplicity and the fact that it can not give to good results, which benefits the demonstration of how empirical risk minimization works. We seek a decision rule as follows:*

$$g(x) = \begin{cases} 1 & \text{if } \ ax_1 + x_2 < b \\ 0 & \text{otherwise.} \end{cases} \tag{9.6}$$

The problem therefore lies within finding the values of the coefficients a and b. To specify the empirical risk in this function we let $\lambda = 0$, use the zero-one

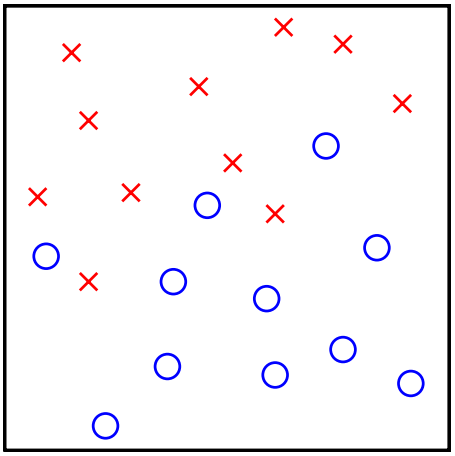


Figure 9.5 Data for Example 9.1.7.

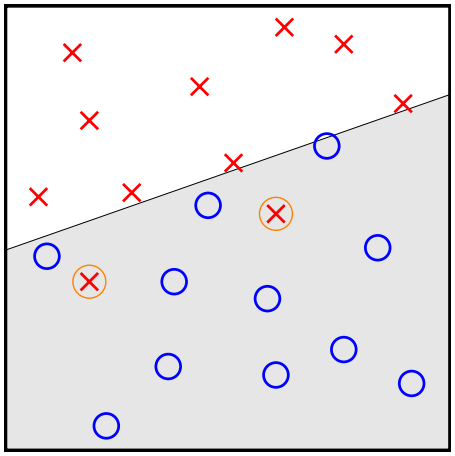


Figure 9.6 Solution for Example 9.1.7.

loss function in (9.2) and restrict $\mathcal{F}$ to the functions of the form (9.6). The problem has now boiled down to finding the line that classifies the points with the least number of errors. As we can see not classifying any of the points wrong will be impossible. By using a computer and varying a and b we can find optimal solutions. One of these are $a = -0.35$, $b = 2.25$, depicted in Figure 9.6. Here we have made two classifications which are wrong, which have been encircled in orange. This is as good as a function of the form in (9.6) can perform for this data set.

Reproducing Kernel Methods

Reproducing kernel methods generalize the simple kernel rule in (9.4), while employing the principles of empirical risk minimization. To be more exact the reproducing kernel methods follow (9.5) using $\mathcal{F} = \mathcal{H}_K$, i.e., the reproducing kernel Hilbert space, or simply RKHS given a kernel $\kappa(\cdot, \cdot)$. We can, given a kernel $\kappa(\cdot, \cdot)$, construct a unique collection of functions $\mathcal{H}_K$ such that

$$\begin{aligned} &\kappa_{x_i} = \kappa(\cdot, x_i) \in \mathcal{H}_K \quad \forall\, x_i \in \mathcal{X}, \\ &\sum_{i=1}^{m} \alpha_i \kappa_{x_i} \in \mathcal{H}_K \quad \forall\, \{\alpha_i\}_{i=1}^{m} \subset \mathbb{R}, \quad m < \infty. \end{aligned} \tag{9.7}$$

If we equip $\mathcal{H}_K$ with an inner product defined by $\langle \kappa_{x_i}, \kappa_{x_j} \rangle_{\mathcal{H}_K} = \kappa(x_i, x_j)$, extend $\mathcal{H}_K$ using linearity to all the functions of the form (9.7), and include the point-wise limits, then $\mathcal{H}_K$ is called an RKHS. From this follows that

$$f(x) = \langle f, \kappa_x \rangle_{\mathcal{H}_K}$$

for all $x \in \mathcal{X}$ and all $f \in \mathcal{H}_K$. This is the reproducing property that has given the name. The norm associated with $\mathcal{H}_K$ can now be denoted by $\|\cdot\|_{\mathcal{H}_K}$.

Let us return to the learning using kernel methods. The inner product structure of $\mathcal{H}_K$ implies the following theorem. The proof complexity goes behyond the scope of this book, but can be found in (Schölkopf and Smola, 2002).

Theorem 9.1.8 (**Representer Theorem**). *The minimizer $g_m^\lambda \in \mathcal{H}_K$ of (9.5) admits a representation of the form*

$$g_m^\lambda(x) = \sum_{i=1}^{m} c_{m,i}^\lambda \kappa(x, x_i) \,, \tag{9.8}$$

for some $c_m^\lambda \in \mathbb{R}^m$.

Theorem 9.1.8 is very important since it highlights that while the optimization in (9.5) is defined over a possibly infinite dimensional Hilbert space, the minimizer must lie in a finite dimensional subspace. It also shows us how the RKHS method generalizes the simple kernel method described in (9.4), owing to the fact that (9.8) reduces to (9.4) for a particular choice of c_m^λ. The significance of the Representer Theorem can perhaps most clearly be seen if we consider the least-square estimation where we can find the solution for c_m^λ in analytic form. This is done in Example 9.1.9.

Example 9.1.9. *We are seeking the function on the form (9.8) that minimizes (9.5) with $l(\cdot,\cdot)$ according to (9.3). In other words finding*

$$f^o = \arg\min_{f \in \mathcal{H}_K} F(f) \,,$$

where we denote

$$F(f) = \frac{1}{m} \sum_{i=1}^{m} \|f(x_i) - y_i\|^2 + \lambda \|f\|_{\mathcal{H}_K}^2 \,.$$

Now we use the optimal function f^o and consider the new function:

$$\tilde{F}(\epsilon) = F\left(f^o + \epsilon \tilde{f}\right) \,,$$

where $\epsilon \in \mathbb{R}$ and $\tilde{f} \in \mathcal{H}_K$. The minimum is obtained with $\epsilon = 0$ and thus we have that

$$\frac{d}{d\epsilon}\tilde{F}(\epsilon)\bigg|_{\epsilon=0} = \frac{d}{d\epsilon}F\left(f^o + \epsilon\tilde{f}\right)\bigg|_{\epsilon=0} = 0, \quad \forall\ \tilde{f} \in \mathcal{H}_K \,.$$

Expanding $F(f)$ *we get* $\left.\frac{d}{d\epsilon}F\left(f^o+\epsilon\tilde{f}\right)\right|_{\epsilon=0} =$

$$
\begin{aligned}
&= \frac{d}{d\epsilon}\left(\frac{1}{m}\sum_{i=1}^{m}\|f^o(x_i)+\epsilon\tilde{f}(x_i)-y_i\|^2+\lambda\|f^o+\epsilon\tilde{f}\|^2_{\mathcal{H}_K}\right)\Bigg|_{\epsilon=0}\\
&= \frac{2}{m}\sum_{i=1}^{m}(f^o(x_i)+\epsilon\tilde{f}(x_i)-y_i)\tilde{f}(x_i)+2\lambda\langle f^o+\epsilon\tilde{f},\tilde{f}\rangle_{\mathcal{H}_K}\Bigg|_{\epsilon=0}\\
&= \frac{2}{m}\sum_{i=1}^{m}(f^o(x_i)-y_i)\tilde{f}(x_i)+2\lambda\langle f^o,\tilde{f}\rangle_{\mathcal{H}_K}=0\,.
\end{aligned}
$$

This is true for all $\tilde{f}\in\mathcal{H}_K$ *and in particular* $\tilde{f}=\kappa_x$ *giving us*

$$
\begin{aligned}
&\frac{1}{m}\sum_{i=1}^{m}(f^o(x_i)-y_i)\kappa(x,x_i)+\lambda\langle f^o,\kappa_x\rangle_{\mathcal{H}_K} =\\
&\qquad\frac{1}{m}\sum_{i=1}^{m}(f^o(x_i)-y_i)\kappa(x,x_i)+\lambda f^o=0\,. \quad\Rightarrow\\
&\qquad f^o=\sum_{i=1}^{m}\frac{1}{\lambda m}(f^o(x_i)-y_i)\kappa(x,x_i)=\sum_{i=1}^{m}c^{\lambda}_{m,i}\kappa(x,x_i)\,.
\end{aligned}
$$

We conclude that the minimizer $f^o\in\mathcal{H}_K$ *of* $F(f)$ *can be represented on the form*

$$f^o(x)=\sum_{i=1}^{m}c^{\lambda}_{m,i}\kappa(x,x_i)$$

and further that the coefficients are implicitly given by

$$
\begin{aligned}
c^{\lambda}_{m,i}&=\frac{1}{\lambda m}(f^o(x_i)-y_i)=\frac{1}{\lambda m}\left(\sum_{j=1}^{m}c^{\lambda}_{m,j}\kappa(x_i,x_j)-y_i\right)\,.\Rightarrow\\
y_i&=\sum_{j=1}^{m}c^{\lambda}_{m,j}\kappa(x_i,x_j)+\lambda m c^{\lambda}_{m,i}\,.
\end{aligned}
$$

This can be written on matrix form as $y=(K+\lambda mI)c^{\lambda}_m$*, giving us*

$$c^{\lambda}_m=(K+\lambda mI)^{-1}y\,. \tag{9.9}$$

Here $K=(k_{ij})$ *is the kernel matrix, with* $k_{ij}=\kappa(x_i,x_j)$*. We thus have proven the Representer theorem for the case of squared error loss function and shown how the coefficients* c^{λ}_m *can be found by solving a system of* m *linear equations.*

The method discussed in Example 9.1.9 is well understood and has been highly successfully implemented in applications ranging from bioinformatics, see (Scholkopf et al., 2003) and (Ben-Hur and Noble, 2005), to hand-written character recognition, see (Bahlmann et al., 2002) and (Zhang et al., 2008).

9.1.2 ARMA-time Series

Auto-Regressive and Moving-Average time series, or short ARMA-time series, is a very versatile method for understanding how natural phenomena work. The strategy requires the phenomena of interest to be bounded and vary around a constant value. This makes the approach perfect for studying phenomena that vary around some equilibrium point. The main focus for ARMA-time series is in predicting how a system will behave in the future given how it has behaved in the past. As it is a time series it uses discrete time samples and relates these different time steps to each other. The ARMA model considers how the system has looked in the past time steps but also how disturbing forces has looked in these steps. The model considers only a certain number of past steps, which may be different for the system state and disturbances parts however.

ARMA-time series can be used for predictions in lots of different systems, e.g., whether, population monitoring, climate, pollution monitoring, economy and different natural phenomena like earthquakes and volcano eruptions. The possibilities are practically infinite and many areas to which ARMA can be applied are probably still waiting to be found. As we shall see many processes that we know to be described by ordinary differential equations, ODEs, can approximately be described by ARMA-time series as well.

Auto-Regressive Model

The AR part of an ARMA process handles how earlier states of the system influence the next state according to

$$d(n) = v(n) + \beta_1 d(n-1) + \beta_2 d(n-2) + \ ... \ + \beta_M d(n-M), \tag{9.10}$$

or in a more compact form

$$d(n) = \sum_{m=1}^{M} \beta_m d(n-m) + v(n), \tag{9.11}$$

where $d(n)$ is the state of the system at time n, β_m are the weights determining the systems behavior, M is the number of time steps back that the model considers, and $v(n)$ is some stochastic white noise, which contains disturbances to the system. If we denote

$$\begin{aligned} u_A(n) &= [d(n-1)\ d(n-2)\ ...\ d(n-M)], \\ \omega_A^o &= \text{col}\{\beta_1, \beta_2, \beta_3, ..., \beta_M\}, \qquad (M \times 1), \end{aligned}$$

where col$\{\cdot\}$ is the column vector containing the given arguments, we can then write

$$d(n) = u_A(n)\omega_A^o + v(n). \tag{9.12}$$

Example 9.1.10. *Here we familiarize ourselves with the concept of an AR-time series. To do this we consider a simple AR(1) process, i.e., where we have* $M = 1$*. We then have*

$$d(n) = \beta d(n-1) + v(n)\,.$$

In this example we are going to consider the development of oil price. To do this we assume that $d(n)$ *is the difference from the average oil price in some arbitrary currency, at month* n*.* $v(n)$ *is the price change originating from disturbances in the oil supply at month* n*. We will restrict restrict ourselves to* $0 < \beta < 1$*, here choosing* $\beta = 0.5$*. We demand a positive value for the model to be feasible and a value less than one for the process to be stationary. Let us say that we start at the equilibrium of* $d(0) = 0$*. We assume that we have a decrease in oil supply at month* $n = 1$ *resulting in* $v(1) = 100$*, could be a oil rig explosion. We hence have*

$$d(1) = \beta d(0) + v(1) = 0.5 \cdot 0 + 100 = 100\,.$$

So at month $n = 1$ *we will have a higher price than usually. This is to expect since we have had a decrease in supply. If we assume no other disturbances, at month* $n = 2$ *we will have*

$$d(2) = \beta d(1) + v(2) = 0.5 \cdot 100 + 0 = 50\,.$$

By this logic the price difference $d(n)$ *will half every month, and return to the initial value after infinite time. This is assuming no further disturbances, of course. This model makes sense, since a decrease in oil supply will discourage investors, giving the system an investor inertia which will increase prices. As the investors gradually return, the price will approach the initial one, assuming no further changes in supply. This behavior can be seen in Figure 9.7.*

Moving-Average Model

The MA part of an ARMA process handles how earlier interference, or disturbances, to the system, $\{u_n\}_{-\infty}^{n}$, influence the current state and is described by

$$d(n) = v(n) + \beta_0 u(n) + \beta_1 u(n-1) + \ \ldots\ + \beta_{(M-1)} u(n-M+1), \quad (9.13)$$

or again in the more compact form

$$d(n) = \sum_{m=0}^{M-1} \beta_m u(n-m) + v(n), \quad (9.14)$$

where $d(n)$ is the state of the system at time n, $u(n)$ the interference at time n, β_m are the weights determining the systems behavior, M is the number

of time steps that interference has an effect on the system, and $v(n)$ is some stochastic white noise, no longer containing the disturbances since these are accounted for. If we denote

$$u_M(n) = [u(n)\ u(n-1)\ ...\ u(n-M+1)],$$
$$\omega_M^o = \text{col}\{\beta_0, \beta_1, \beta_2, ..., \beta_{(M-1)}\}, \qquad (M \times 1),$$

we can then write

$$d(n) = u_M(n)\omega_M^o + v(n). \tag{9.15}$$

Example 9.1.11. *Here we familiarize ourselves with the concept of an MA-time series. To do this we consider a simple MA process, where we have* $M = 2$*. We have*

$$d(n) = \beta_0 u(n) + \beta_1 u(n-1) + v(n)\ .$$

In this example we are going to consider the development of sales of big pack ice cream. To do this we assume that $d(n)$ *is the difference from the average sold number of big packs at a particular store on day* n*.* $u(n)$ *is the on day* n *change in temperature from the day before.* $v(n)$ *is some white noise parameter, which we here neglect (*$v(n) = 0\ \ \forall\ n$*). We will use* $\beta_0 = 100$ *and* $\beta_1 = -50$*. We start at the equilibrium of* $d(0) = 0$ *and* $u(0) = 0$*. We assume that we have an increase in temperature on day* $n = 1$*,* $u(1) = 3$*. We hence have*

$$d(1) = \beta_0 u(1) + \beta_1 u(0) = 100 \cdot 3 - 50 \cdot 0 = 300\ .$$

So on day $n = 1$ *we will have more sales than usually. This is to expect since we have had an increase in temperature, making ice cream more tempting. If we assume no further changes in temperature, on day* $n = 2$ *we will have*

$$d(2) = \beta_0 u(2) + \beta_1 u(1) = 100 \cdot 0 - 50 \cdot 3 = -150\ .$$

The sales are now lower than usual, which can be explained by the higher sales the day before. Because people bought ice cream yesterday, most have not finished the big pack the next day and will not need to buy another that day. As the series looks now the sales would be back at usual in the next day if no further temperature changes occur. This behavior can be seen in Figure 9.8. By involving a higher order MA process the model could consider more steps, and thus have a longer lasting outcome.

The Complete ARMA Model

The AR and MA models are very similar and can be combined in the ARMA model. With $u_n = [u_M(n)\ u_A(n)]$ and $\omega = \text{col}\{\omega_M, \omega_A\}$ we get the complete model as

$$d(n) = u_n\omega^o + v(n), \tag{9.16}$$

where u_n is a row vector, of length $M = M_{\mathrm{AR}} + M_{\mathrm{MA}}$, containing all the information on the history of the system and the interference it has experienced and ω is a column vector of equal length containing the system model. $v(n)$ is now though to contain factors that have not been considered and measurement noise. If one knows the ω of the system one knows exactly how it works and can therefore predict future behavior based on the knowledge of u_n.

Example 9.1.12. *Here we familiarize ourselves with the concept of an ARMA-time series. To do this we consider a simple ARMA(1,1) process, i.e., where we have $M_{\mathrm{AR}} = 1$, $M_{\mathrm{MA}} = 2$. We then have*

$$d(n) = \beta_0 u(n) + \beta_1 u(n-1) + \beta_2 d(n-1) + v(n) \,.$$

In this example we are going to consider the development of monthly sales at a company store. To do this we assume that $d(n)$ is the difference from the average sales during month n. $u(n)$ is the difference in advertisement for the company during month n as opposed to the usual. $v(n)$ is some white noise parameter, which we here neglect ($v(n) = 0 \quad \forall\, n$). We will use $\beta_0 = 100$, $\beta_1 = 70$ and $\beta_2 = 0.5$. We start at the equilibrium of $d(0) = 0$ and $u(0) = 0$. We assume that we have an increased advertisement during month $n = 1$ giving us $u(1) = 10$. We hence have

$$d(1) = \beta_0 u(1) + \beta_1 u(0) + \beta_2 d(0) = 100 \cdot 10 + 70 \cdot 0 + 0.5 \cdot 0 = 1000 \,.$$

So during month $n = 1$ we will have more sales than usually. This is to expect since we have advertised more for the products we sell. If we assume that all future advertisement is at the usual level, giving $u(n > 1) = 0$, on day $n = 2$ we will have

$$d(2) = \beta_0 u(2) + \beta_1 u(1) + \beta_2 d(1) = 100 \cdot 0 + 70 \cdot 10 + 0.5 \cdot 1000 = 1200 \,.$$

The sales of month $n = 2$ are slightly higher than those of month $n = 1$. This can be explained by the still lasting effects of the advertisement, the MA-part, in combination with the loyalty that costumers get to the store after shopping there last month, the AR-part. The coming months, the effects of the advertisement will be gone, but the loyalty effect will keep costumers buying goods, keeping the sales higher, although decreasingly, for a long time. This behavior can be seen in Figure 9.9.

Applied to Differential Equations

Many phenomena are mathematically described by differential equations and it would therefore be very useful if ARMA-series could be applied to these phenomena. By using discrete approximations for the derivative one

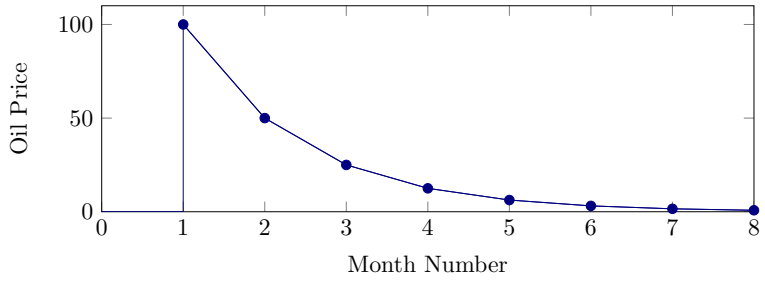


Figure 9.7 The process from Example 9.1.10.

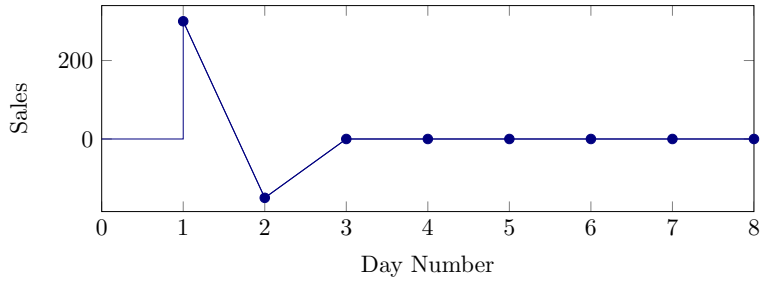


Figure 9.8 The process from Example 9.1.11.

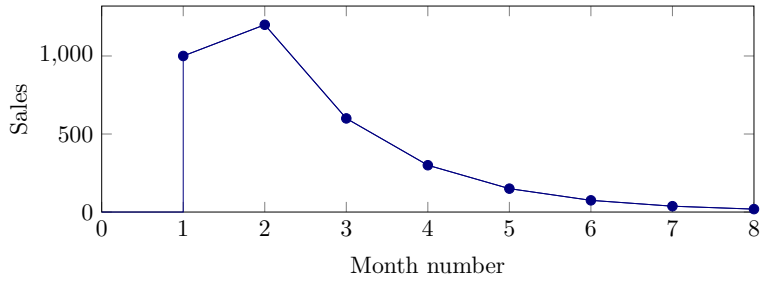


Figure 9.9 The process from Example 9.1.12.

ends up with terms that can be fit into the ARMA-framework. Here we merely consider the simple *Euler Backwards* method of approximating the derivative whereas other more accurate methods, e.g., *Tustin's Formula*, could certainly be applied. Here however we just want to show that ARMA-time series and differential equations can be two means of describing the same thing.

The *Euler Backwards* method for the first and second derivative are given by:

$$\frac{D}{Dt}d(n) \approx \frac{1}{T}\left(d(n) - d(n-1)\right) \ , \tag{9.17}$$

$$\frac{D^2}{Dt^2}d(n) \approx \frac{1}{T^2}\left(d(n) - 2d(n-1) + d(n-2)\right) \ . \tag{9.18}$$

Using these equations many differential equations can be written on the ARMA-form. In Example 9.1.13 we show how an electric circuit can be described by ARMA-time series through converting an ODE.

Example 9.1.13. *Consider a circuit which consists of an inductor, a resistor, a capacitor and a voltage source. We denote the resistance, the inductance and the capacitance of the circuit with R, L and C, respectively. We also denote the electromotive force from the voltage source by $E(t)$. Then the current through the circuit, $I(t)$, follows*

$$L\frac{D^2}{Dt^2}I(t) + R\frac{D}{Dt}I(t) + \frac{1}{C}I(t) = \frac{D}{Dt}E(t)\,.$$

Now we use (9.17) and (9.18) to convert the equation above to a time series, assuming short enough time step T:

$$\begin{aligned}
&L\frac{1}{T^2}\left(I(n) - 2I(n-1) + I(n-2)\right) + R\frac{1}{T}\left(I(n) - I(n-1)\right) + \frac{1}{C}I(n)\\
&= \frac{1}{T}\left(E(n) - E(n-1)\right) \quad \Rightarrow\\
&\underbrace{\left(\frac{L}{T^2} + \frac{R}{T} + \frac{1}{C}\right)}_{A} I(n) - \underbrace{\left(2\frac{L}{T^2} + \frac{R}{T}\right)}_{B} I(n-1) + \underbrace{\frac{L}{T^2}}_{D} I(n-2)\\
&= \frac{1}{T}E(n) - \frac{1}{T}E(n-1) \quad \Rightarrow\\
&I(n) = \frac{1}{AT}E(n) - \frac{1}{AT}E(n-1) + \frac{B}{A}I(n-1) - \frac{D}{A}I(n-2)
\end{aligned}$$

With

$$u(n) = [E(n)\ E(n-1)\ I(n-1)\ I(n-2)], \quad \omega^o = \operatorname{col}\left\{\frac{1}{AT}, -\frac{1}{AT}, \frac{B}{A}, -\frac{D}{A}\right\},$$

we have

$$I(n) = u(n)\omega^o + v(n)\,,$$

where we have added a white noise term for measurement uncertainties. This is on the ARMA-time series form and hence we have shown how an ODE can be transformed into an ARMA-time series. Note that ω^o here has two terms pertaining to the MA-part and another two pertaining to the AR-part.

9.1.3 Optimization in Learning Algorithms

A very common problem that arises in leaning is optimization, i.e., finding some parameter that optimizes some problem. As we shall see, the problem we are interested in through this chapter is minimizing a function.

This has already been seen in equation (9.5). Generally we formulate this as the problem of finding the parameter f that generates the lowest value of some cost function $J(f)$. The problem can be stated as finding the optimal f^o

$$f^o := \arg\min_{f \in \mathcal{F}} J(f)\,, \tag{9.19}$$

where $\mathcal{F}$ is some space which we constrain f to be within. This can be anything from Hilbert function space to the simple $\mathbb{R}$.

Steepest Decent Method

One very useful way of solving minimizing optimization problems is the so called Steepest Decent or Gradient Decent Method. It uses that the gradient of a function points in the direction where the function grows fastest. Thus the opposite direction is the direction of steepest decent. By always following the path of steepest decent the algorithm finds a minimum, given a short enough step size. If we further know that we have a convex problem the method will find the global minimum. The algorithm is an iterative process according to

$$x(n+1) = x(n) - \mu_n \nabla_x J(x(n)), \tag{9.20}$$

where $x(n)$ is the variable of the cost function J, μ_n is the step size at time n and ∇_x is the gradient with respect to x.

Mean Square Cost Function

The cost function J can be constructed in many ways but a very common one, and the one we will mostly be using in this chapter, is the Mean Square Error, MSE, cost function. It takes the expected squared error as the cost to minimize. In general, the variable which we want to find can be a function, a vector, or even a scalar. When constructing the error we assume that we have some measurement d, some parameters x prescribed to the measurement and an estimate of the sought parameter f. In general the sought quantity f and the prescribed parameters x are related to the measurement d by a function $h(f, x)$. This function predicts the measurement based on the model parameter f and the prescribed parameters x. This now gives us the error

$$e(f) := h(f, x) - d$$

The prescribed parameters could for example be the training data input in the kernel approach or the vector u in the ARMA method. The function $h(f, x)$ is usually simple as in the kernel approach, where we have $h(f, x) = f(x)$. More discussion on this will follow. Anyhow we now get the MSE as

$$\text{MSE} := J(f) := \mathbb{E}\left\{\|e(f)\|^2\right\} = \mathbb{E}\left\{\|h(f, x) - d\|^2\right\}. \tag{9.21}$$

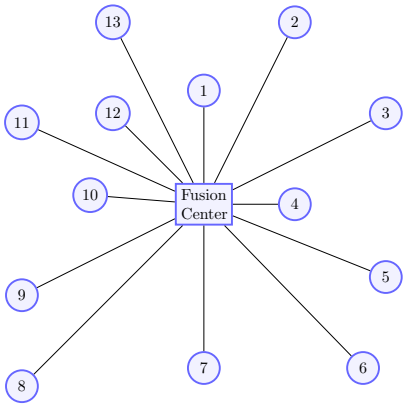


Figure 9.10 An example of a Star Network with a fusion center.

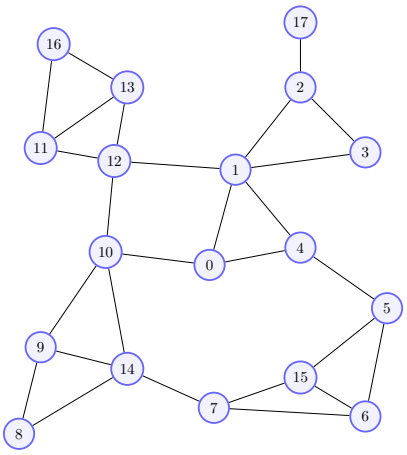


Figure 9.11 An example of a general network topology where each node acts as a peer.

The MSE cost function has the nice property of being a convex function making optimization involving it easy to solve.

9.2 Learning in WSNs

The usage of WSNs in learning problems opens for new possibilities. WSNs allow for information gathering over a wide area and the network can be made very robust. In this section we will look at how we can use the methods from last section in WSNs. In the same way that WSNs has certain advantages they also have their downsides, one of which is the tight constraints on energy consumption. This makes the communication in the network costly and hence makes many learning methods unfit in WSNs. Depending on which network topology the network has different techniques can be used and in this section we will discuss both the star topology and general network topology. The star network has a central node, known as the fusion center, to which all other nodes are connected and no other connections exist. The general network has no predefined structure but generally connects only nodes that are close enough to each other. Examples of these types of network can be seen in Figure 9.10 and Figure 9.11. We will look at methods that try to communicate as little as possible to avoid depleting the nodes' energy sources.

9.2.1 Star Topology

When the network has the shape of a star and thus has a fusion center, the central node, the network nodes could simply send the measurements to the fusion center. At the fusion center all the calculations in the used learning algorithm could be performed. This approach however does not take the restrictions on communication into account and in practice the limited energy and bandwidth makes this simple approach unfeasible. Furthermore it does not use the available computation capabilities that all the nodes posses. Thus we would like to investigate how local calculations can lower the need for communication and still give good learning results.

Lower Limit on Communication

In this section we show that good learning performance can be achieved by reduced communications. Here we consider kernel methods in general but since we can see ARMA-time series as a special case of the estimation problem in supervised learning the results studied in this section hold there as well.

We suppose that the restrictions on power consumption and bandwidth limit each sensor to only send one bit of information in each time step. In each time step the fusion center sends a request for information to the nodes which each has the choice to respond with one bit, i.e., 0 or 1, or to abstain from sending anything. Under these assumptions, we are interested in studying if is possible to achieve universally consistent learning. As stated by the following theorem this is possible.

Theorem 9.2.1 (Classification and Estimation with Abstention)**.** *Suppose that the data from the sensors,* $S_m = \cup_{j=1}^{K} S_m^j$ *are i.i.d. with* $(X_i, Y_i) \sim \boldsymbol{P}_{XY}$ $\forall i \in \{1, ..., K\}$*, and the individual sensors know* K*, i.e., the size of the network. Then, in binary classification under zero-one loss, and in estimation under squared-error criterion, there exist a sensor decision algorithm and a fusion rule that enable universally consistent distributed learning with abstention.*

Theorem 9.2.1 is proven by construction and the proof can be found in (Predd et al., 2006a). By proven by construction we mean that given the constraints a sensor decision algorithm and a fusion rule are found that are provable universally consistent. The decision rule that was proven to be universally consistent in the classification case is discussed in more detail in Example 9.2.4.

As the problem is formulated now the sensors are able not to respond. If each sensor has to respond however, giving them two choices instead of three will this still be possible? The two following theorems address this issue.

Theorem 9.2.2 (Classification without Abstention)**.** *Suppose that the data from the sensors,* $S_m = \cup_{j=1}^{K} S_m^j$*, are i.i.d. with* $(X_i, Y_i) \sim \boldsymbol{P}_{XY}$ $\forall i \in \{1, ..., K\}$*, and the individual sensors know* K*, i.e., the size of the network. Then in binary classification under zero-one loss there exist sensor decision algorithm and a fusion rule that enable universally consistent distributed learning without abstention.*

Theorem 9.2.2 is in the same way as Theorem 9.2.1 proven by construction in (Predd et al., 2006a). In Example 9.2.5 this case is considered. To tackle the estimation problem without abstention we first introduce an assumption.

Assumption 1. Assume that the considered fusion rule is invariant to the order in which bits are received from the network. Further assume that for all $x \in \mathcal{X}$ the decision rule is Lipschitz continuous in the average Hamming distance, i.e., if we order the received bits in a vector $b \in \{0,1\}^m$ and denote the decision rule $g_m(x, b)$, there exists a constant C so that

$$\|g_m(x, b_1) - g_m(x, b_2)\| \leq C \frac{1}{m} \sum_{i=1}^{m} \|b_{1i} - b_{2i}\|$$

for every $b_1, b_2 \in \{0,1\}^m$.

With Assumption 1 we are now ready for the theorem about estimation without abstention.

Theorem 9.2.3 (Estimation without Abstention)**.** *Suppose that the data from the sensors,* $S_m = \cup_{j=1}^{K} S_m^j$ *are i.i.d. with* $(X_i, Y_i) \sim \boldsymbol{P}_{XY}$ $\forall i \in \{1, ..., K\}$*, that the individual sensors know* K*, i.e., the size of the network and that the fusion rule satisfies Assumption 1. Then, for any sensor decision algorithm that obeys the constraints of distributed learning without abstention, there does not exist a regular fusion rule that is consistent for every distribution* $\boldsymbol{P}_{XY}$ *with* $\mathbb{E}\{Y^2\} < \infty$ *under the squared-error criterion.*

Theorem 9.2.3 is proved in (Predd et al., 2006a) through a counterexample, and thereby establishes the impossibility of universally consistency in distributed regression without abstention under Assumption 1, which implies a restricted but reasonable class of wireless sensor networks.

Example 9.2.4. *Here we discuss a universally consistent decision rule for the classification with abstention. We assume that each sensor has one training data sample. When the fusion center wants to know how to classify* $x \in X$ *it sends this value to all the nodes. These in turn use their training sample,* (x_k, y_k)*, and* x *to decide whether to respond, with 0 or 1, or to abstain. The nodes decide this by using the naive kernel in Table 9.1 here denoted* $\kappa(\cdot, \cdot)$*.*

Let us denote the response from node k as δ_k. We can then formulate the response as

$$\delta_k(x) = \begin{cases} y_k & \text{if} \quad \kappa(x, x_k) = 1 \\ \text{abstain} & \text{otherwise} \end{cases} .$$

The fusion center then fuses these responses to get the prediction y according to

$$g(x) = \begin{cases} 1 & \text{if} \quad \sum_{i \in \mathcal{R}} \delta_i \geq \frac{1}{2} \|\mathcal{R}\| \\ 0 & \text{otherwise} \end{cases} .$$

Here $\mathcal{R}$ denotes the set of nodes that did respond and $\|\mathcal{R}\|$ denotes the size of this set, i.e., the number of responding nodes. The decision rule is simply the majority vote of the nodes that are close enough, determined by r_m in the naive kernel, to x to get to vote. This decision rule is actually the same as the one in (9.4) given that we round the result. Now for the decision rule to be universally consistent we have to restrict ourselves to the case where

$$r_m \to 0 \quad \text{and} \quad (r_m)^d K \to \infty \quad \text{as} \quad K \to \infty ,$$

where d is the dimension of the input space X. If these conditions hold we have a universally consistent learning rule.

Example 9.2.5. *Here we discuss a universally consistent decision rule for the classification without abstention. We use the same assumptions as in Example 9.2.4. We only remove the possibility of abstaining from responding, forcing the nodes to send either 0 or 1 back to the fusion center. This time we include a function that on random chooses either 0 or 1 with equal probability. We call this function $\theta_{0/1}$, and have $P(\theta_{0/1} = 1) = P(\theta_{0/1} = 0) = 0.5$. With this we get*

$$\delta_k(x) = \begin{cases} y_k & \text{if} \quad \kappa(x, x_k) = 1 \\ \theta_{0/1} & \text{otherwise} \end{cases} .$$

The fusion center then fuses these responses to get the prediction y according to

$$g(x) = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{K} \delta_i \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} .$$

Here we let the close nodes respond their training data output but let the other nodes make an unqualified guess. With many of these unqualified responses they will cancel each other out. Now for the decision rule to be universally consistent we have to restrict ourselves to the case where

$$r_m \to 0 \quad \text{and} \quad (r_m)^d \sqrt{K} \to \infty \quad \text{as} \quad K \to \infty ,$$

where again d is the dimension of X. If these conditions hold we have a universally consistent learning rule.

9.2.2 General Topology

In this section we consider the case when our network has general structure. We mainly discuss two ways of handling general topology WSNs, using a cyclic path through the network or using diffusion methods. The methods are very general and can be applied to almost any network. The methods also do not assume there to be any fusion center and thus gets rid of the central link that requires long range communication and gives the network a weak link. The methods are first developed in general and then applied to the two cases of kernel methods and ARMA-time series in the last parts of this section. Throughout this chapter we will assume that the networks we are working with are connected, i.e., the graph representing the network is connected which means that for each pair of nodes in the graph there is a path connecting them.

As we have seen earlier the concept of learning often boils down to finding a parameter that minimizes a cost function $J(\cdot)$. This is the case in this section and thus we discuss the network cost function below.

Network Cost Function

Because of the network structure the cost function $J(f)$, to which we want to find the minimizer f, usually naturally splits up in a sum of individual cost functions $J_k(f)$ over all the K sensors:

$$J(f) = \sum_{k=1}^{K} J_k(f)\,. \tag{9.22}$$

This is not always the case, but we will in this chapter only consider this situation. This restriction is in practice not very restricting since terms that does not naturally belong to one of the sensors can be split between them or just be assorted to one of them.

The network's goal is to estimate the function or variable f, which minimizes the cost function $J(f)$, and we can therefore use the previously described steepest decent method in (9.20). If f_n denotes the n-th iterative approximation of f we have:

$$f_{n+1} = f_n - \frac{\mu_n}{2}\nabla_f J(f_n), \tag{9.23}$$

but from (9.22) we get

$$\nabla_f J(f) = \nabla_f \sum_{k=1}^{K} J_k(f) = \sum_{k=1}^{K} \nabla_f J_k(f) = \sum_{k=1}^{K} H_k(f),$$

where we define the new function

$$H_k(f) \triangleq \nabla_f J_k(f). \tag{9.24}$$

We then get

$$f_{n+1} = f_n - \frac{\mu_n}{2} \sum_{k=1}^{K} H_k(f_n)\,. \tag{9.25}$$

Thus we have modified the steepest decent algorithm to the case where the cost function splits up into a sum.

Local MSE Cost Functions

A commonly used and simple local cost function, i.e., $J_k(\cdot)$, is just the local MSE in (9.21). We will denote this choice of local cost function as $J_k^{\text{loc}}(\cdot)$, where the superscript symbolizes that only *local* information is used. This gives us

$$J_k^{\text{loc}}(f) = \text{MSE}_k = \mathbb{E}\left\{\|h_k(f) - d_k\|^2\right\}, \tag{9.26}$$

Using $J_k(f) = J_k^{\text{loc}}(f)$ in (9.22) gives us

$$J(f) = \sum_{k=1}^{K} J_k^{\text{loc}}(f) = \sum_{k=1}^{K} \mathbb{E}\left\{\|h_k(f) - d_k\|^2\right\}. \tag{9.27}$$

If we use the $J_k^{\text{loc}}(f)$ in (9.26) and (9.24) we have

$$H_k^{\text{loc}}(f) = \nabla_f \mathbb{E}\left\{\|h_k(f) - d_k\|^2\right\} = \mathbb{E}\left\{2(h_k(f) - d_k)\frac{dh_k(f)}{df}\right\}. \tag{9.28}$$

As stated now equation (9.28) is a bit abstract, but the form of $h(\cdot,\cdot)$ is usually very simple as we shall see in Section 9.2.3 and 9.2.4, when we apply this equation in the specific cases of kernel methods and ARMA-time series.

Cyclic Path Method

Notice how, if we use $H_k(f_n) = H_k^{\text{loc}}(f_n)$, each update of f_n in (9.25) involves a sum incorporating one term with only local data from each sensor. This motivates the following algorithm:

$$\begin{aligned}
&\psi_0^n \leftarrow f_{n-1},\\
&\psi_k^n \leftarrow \psi_{k-1}^n - \frac{\mu_n}{2} H_k^{\text{loc}}(f_{n-1}), \qquad k = 1,\ \ldots\ , K,\\
&f_n \leftarrow \psi_K^n,
\end{aligned} \tag{9.29}$$

with $H_k^{\text{loc}}(f_n)$ as in (9.28). If there exists a cyclic path through the network, as in Figure 9.12, this algorithm can be implemented by letting each sensor compute its estimate, ψ_k^n, from its measured value and the preceding sensor's estimate, ψ_{k-1}^n. One problem is that (9.29) requires all sensors to know

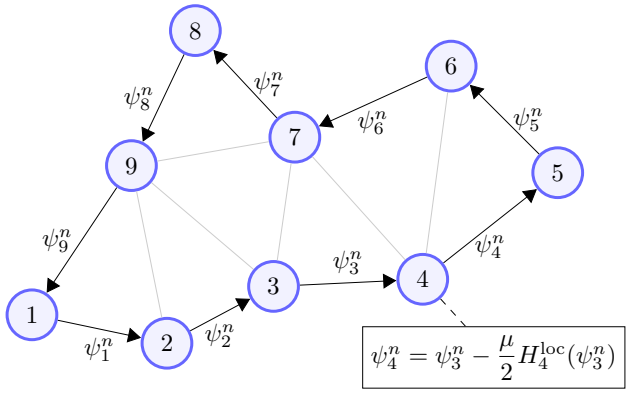


Figure 9.12 A small network using the incremental cyclic path method in (9.30). The arrows represent information flow and the gray lines show possible connections which are not used in this algorithm.

the value of f_n, which enters into $H_k^{\text{loc}}(\cdot)$. To remedy this problem one can instead of the actual value of f_n use the previous local estimate ψ_{k-1}^n. This type of estimate is known as an incremental solution. We also allow the step size to vary with k, instead giving

Incremental Cyclic Path Method

$$\begin{aligned} &\psi_0^n \leftarrow f_{n-1}, \\ &\psi_k^n \leftarrow \psi_{k-1}^n - \frac{\mu_{k,n}}{2} H_k^{\text{loc}}(\psi_{k-1}^n), \qquad k = 1, \ \ldots, K, \\ &f_n \leftarrow \psi_K^n. \end{aligned} \tag{9.30}$$

In Figure 9.12 the algorithm is depicted for a small network consisting of nine nodes. How well (9.30) performs depends on what model is used and we will later in this chapter discus this for kernel methods and ARMA-time series.

A More General Cost Function

The local MSE cost function in (9.26) is in many cases a very good choice for $J_k(f)$, but sometimes a more flexibility is required. Thus we here consider a more general cost function inspired by equation (9.5). Here we define the complete function and seeks local ones that fulfill (9.22) and not the other way around as in the case of the local MSE cost function. Equation (9.5) has a loss function $l(\cdot)$ which we here take as the MSE function in (9.21) giving us:

$$J(f) = \sum_{k=1}^{K} \mathbb{E}\left\{\|h_k(f) - d_k\|^2\right\} + \lambda \|f\|_{\mathcal{F}}^2,$$

which can be rewritten as

$$J(f) = \sum_{k=1}^{K} \left[\mathbb{E} \left\{ \|h_k(f) - d_k\|^2 \right\} + \lambda_k \|f\|_{\mathcal{F}}^2 \right], \tag{9.31}$$

as long as $\sum_{k=1}^{K} \lambda_k = \lambda$. Now imagine that each sensor not only gets information from one other sensor, as in the cyclic method, but from all the sensors of the set of neighbors, $\mathcal{N}_k$, defined as:

$$\mathcal{N}_k = \{\text{the set of nodes connected to } k \text{ including itself}\}. \tag{9.32}$$

Now we want to find a $J_k(f)$ that satisfies (9.22), given the $J(f)$ in (9.31), but that incorporates the information from the other sensors in $\mathcal{N}_k$. To achieve this goal let us first consider the set of non-negative set of coefficients $\{c_{kl}\}$ that for all values of k satisfy:

$$c_{kl} \geq 0, \quad \sum_{l=1}^{K} c_{kl} = 1, \quad \text{and} \quad c_{kl} = 0 \quad \text{if} \quad l \notin \mathcal{N}_k \,. \tag{9.33}$$

If we collect the coefficients $\{c_{kl}\}$ in a matrix $C = \{c_{kl}\}$ the conditions in (9.33) implies that

$$C\mathbb{1} = \mathbb{1}, \tag{9.34}$$

where $\mathbb{1}$ is the $(K \times 1)$ vector defined as

$$\mathbb{1} \triangleq [1, 1, ..., 1]^T \,. \tag{9.35}$$

This makes C a right stochastic matrix. If the restrictions in (9.33) are followed the values of $\{c_{kl}\}$ can be chosen freely. There are a lot of different ways of doing this and by choosing a good way the performance will be better. As we shall see this is the same requirements that we have for an other set of coefficients $\{a_{kl}\}$. For this reason the choices presented for them will work here as well. So keep this in mind when reading the part called *Combination Rules for Diffusion Methods.*

Now using the coefficients in (9.33) we achieve a $J_k(f)$ that satisfies (9.22), given the $J(f)$ in (9.31), through:

$$J_k^{\text{gen}}(f) = \sum_{l \in \mathcal{N}_k} c_{lk} \, \mathbb{E} \left\{ \|h_l(f) - d_l\|^2 \right\} + \lambda_k \|f\|_{\mathcal{F}}^2 \,.$$

Where the superscript symbols that this local cost function is of the *general* kind. Note the change in the order of the subscript on c. Remembering equation (9.26) we can write this:

$$J_k^{\text{gen}}(f) = \sum_{l \in \mathcal{N}_k} c_{lk} J_l^{\text{loc}}(f) + \lambda_k \|f\|_{\mathcal{F}}^2 \,. \tag{9.36}$$

Here it is worth to note that if we choose $c_{lk} = \delta_{lk}$, or equivalently $C = I$, and $\lambda_k = 0 \ \ \forall \ \ k$ we have that (9.36) reduces to (9.26), where δ_{lk} is the Kronecker delta and I the identity matrix.

The equation of (9.36) can be motivated by considering the cost function in (9.31) and using (9.26) and (9.33):

$$\begin{aligned} J(f) &= \sum_{k=1}^{K} \left[J_k^{\text{loc}}(f) + \lambda_k \|f\|_{\mathcal{F}}^2 \right] = \sum_{k=1}^{K} \left[\left(\sum_{l=1}^{K} c_{kl} \right) J_k^{\text{loc}}(f) + \lambda_k \|f\|_{\mathcal{F}}^2 \right] \\ &= \sum_{k=1}^{K} \sum_{l=1}^{K} c_{kl} J_k^{\text{loc}}(f) + \lambda \|f\|_{\mathcal{F}}^2 = \sum_{l=1}^{K} \sum_{k=1}^{K} c_{kl} J_k^{\text{loc}}(f) + \lambda \|f\|_{\mathcal{F}}^2 \\ &= \sum_{l=1}^{K} \left[\sum_{k=1}^{K} c_{kl} J_k^{\text{loc}}(f) + \lambda_l \|f\|_{\mathcal{F}}^2 \right]. \end{aligned}$$

By now switching the index labeling $l \leftrightarrow k$ and using that $c_{kl} = 0$ if $l \notin \mathcal{N}_k$ we arrive at:

$$J(f) = \sum_{k=1}^{K} \left[\sum_{l \in \mathcal{N}_k} c_{lk} J_l^{\text{loc}}(f) + \lambda_k \|f\|_{\mathcal{F}}^2 \right] = \sum_{k=1}^{K} J_k^{\text{gen}}(f) \,,$$

where we have defined $J_k^{\text{gen}}(f) \triangleq \sum_{l \in \mathcal{N}_k} c_{lk} J_l^{\text{loc}}(f) + \lambda_k \|f\|_{\mathcal{F}}^2$, which is as in (9.36).

Now with the new cost function of (9.36) we can calculate the new $H_k^{\text{gen}}(f)$ using (9.36) according to (9.24):

$$\begin{aligned} H_k^{\text{gen}}(f) &= \nabla_f \left(\sum_{l \in \mathcal{N}_k} c_{lk} J_l^{\text{loc}}(f) + \lambda_k \|f\|_{\mathcal{F}}^2 \right) \\ &= \sum_{l \in \mathcal{N}_k} c_{lk} \nabla_f J_l^{\text{loc}}(f) + \lambda_k \nabla_f \|f\|_{\mathcal{F}}^2 = \sum_{l \in \mathcal{N}_k} c_{lk} H_l^{\text{loc}}(f) + 2\lambda_k f \quad \Rightarrow \\ H_k^{\text{gen}}(f) &= \sum_{l \in \mathcal{N}_k} c_{lk} H_l^{\text{loc}}(f) + 2\lambda_k f \,, \end{aligned} \tag{9.37}$$

where we have H_l^{loc} according to (9.28).

Diffusion Methods

We previously showed how the cyclic patch method works in principle. The idea was that each node only received data from a single node and sent data to yet another single node. This approach to the distributed learning problem is in some cases a very good choice. The method is very energy efficient, only requiring the nodes to send and receive from short distances

once per time step. The communications always happen the same way making them easier to optimize as well. Thus this algorithm is very good when energy constraints are tight. However, it has some limiting factors making it unsuitable for general situations. First, the algorithm requires that there is a cyclic path through the network, which might not be the case, at least not that makes sense to implement. We do not want the distances to be large between cyclic neighbors, and if it exceeds the average distance of neighboring nodes too much the algorithm does not make much sense. Secondly the cyclic path makes the network very vulnerable to node failures, since one failure ruins the cyclic path. Thirdly the algorithm requires the nodes to wait for all the other nodes between updates and this limits how fast the algorithm can run. These arguments combined gives strong incentive to search for better methods to use when these issues produce problems.

In most networks, the topology allows for the nodes to communicate with many other nodes and if this is not prohibited by energy restraints we should take advantage of this possibility. If the communication is omnidirectional there might not be more energy exhausting to communicate with all neighbors than it is with one of them. We can benefit from the expanded communication by instead of at node k using the estimate from node $k-1$, as in (9.30), use a weighted average estimate from the nodes of the set $\mathcal{N}_k$ introduced in (9.32). This can be done by:

$$\psi_k^n = \sum_{l \in \mathcal{N}_k} a_k \psi_l^{n-1} \quad \text{with} \quad \sum_{l \in \mathcal{N}_k} a_{kl} = 1 \ \ \forall \ k. \tag{9.38}$$

By using (9.38) as the the local estimate, f_n, and iterate the equation (9.23) once for all nodes in each time step using their local cost functions, J_k which here is not specified, we get a so called diffusion method, or more specifically the Combine-Then-Adapt diffusion algorithm, or short CTA. This method can be described as:

CTA diffusion

$$\begin{aligned} \phi_k^n &\leftarrow \sum_{l \in \mathcal{N}_k} a_{kl} \psi_l^{n-1}, \\ \psi_k^n &\leftarrow \phi_k^n - \frac{\mu_{n,k}}{2} H_k(\phi_k^n)\,. \end{aligned} \tag{9.39}$$

Here the cyclic update has been removed, instead in all time steps n all sensors performs the two step process depicted above. This makes it possible for all nodes to perform updates at the same time and significantly reduces the idle waiting time. By switching the step order in the CTA algorithm we get the Adapt-Then-Combine version as

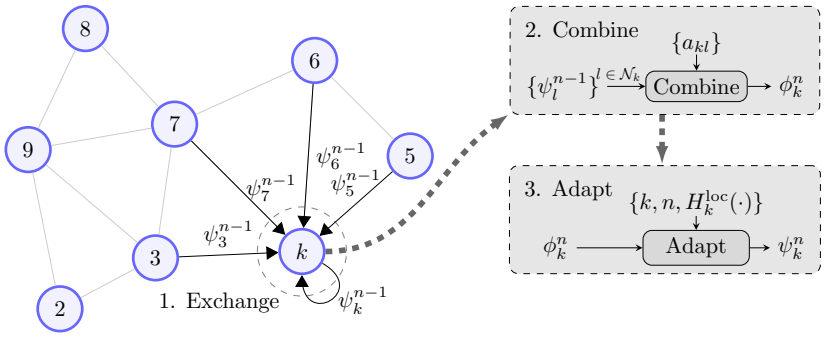


Figure 9.13 The CTA algorithm using local adaptive data. There are three ordered steps: Exchange, Combine and Adapt. All nodes perform this process in every time step, here depicted for node k.

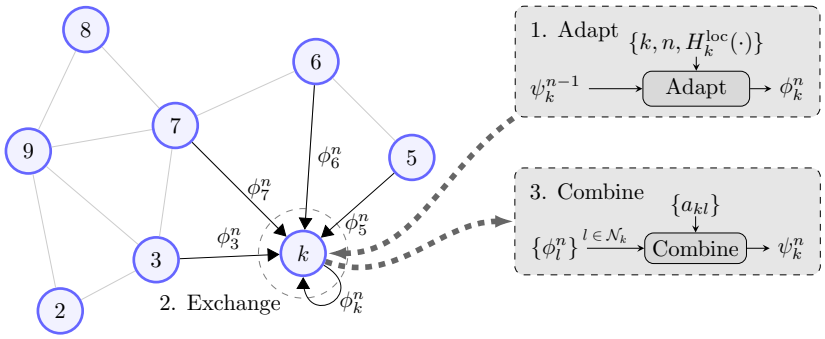


Figure 9.14 The ATC algorithm using local adaptive data. There are three ordered steps: Adapt, Exchange and Combine. All nodes perform this process in every time step, here depicted for node k.

ATC diffusion

$$\begin{aligned}\phi_k^n &\leftarrow \psi_k^{n-1} - \frac{\mu_{n,k}}{2} H_k(\psi_k^{n-1}),\\ \psi_k^n &\leftarrow \sum_{l\in\mathcal{N}_k} a_{kl}\phi_l^n.\end{aligned} \tag{9.40}$$

The CTA and ATC algorithms can be seen depicted, for the case where $H_k(\cdot) = H_k^{\text{loc}}(\cdot)$, in Figure 9.13 and 9.14 respectively. If we were to use the general $H_k^{\text{gen}}(\cdot)$, we would have to include another exchange stage before the adapt state, where we share information of the local data.

A few comments are in order. The CTA and ATC methods are very similar and the question is whether they perform equally well. We will look more closely at this question in Section 9.2.5, but we can mention here that the ATC method generally perform better. This can be intuitively understood

by noting that in ATC the adaption step is performed first incorporating new data before sharing its estimate. This has the effect that the ATC method always works with more recent data than the CTA.

Motivation for Diffusion Methods

The algorithms in (9.40) and (9.39) can be more formally motivated by looking at the global cost function at one of the nodes. As described by (9.22) the global cost function, $J(\cdot)$ for the entire network is a sum over the sensor's individual cost functions. We can rewrite the equation as

$$J(f) = J_k(f) + \sum_{\substack{l=1 \\ l \neq k}}^{K} J_l(f)\,.$$

Note that we have not yet limited what the sought minimizer f is. It could, e.g., be a function, vector or scalar. We do however assume that it is contained in the set $\{\mathcal{F}\}$ and has a definition of norm $\|\cdot\|_{\mathcal{F}}^2$. If we relax the global cost function, $J(f)$, to the neighborhood of k, we obtain

$$J_k^{\mathcal{N}}(f) = J_k(f) + \sum_{l \in \mathcal{N}_k/\{k\}} J_l(f)\,.$$

By now approximating $J_l(f) \approx b_{kl}\|f - f_l^o\|_{\mathcal{F}}^2$, where $f_l^o = \arg\min_{f\in\mathcal{F}}(J_l(f))$, and introduce the regularization parameter, $\delta > 0$, we get the more convenient form

$$J_k^{\mathcal{N}}(f) = J_k(f) + \delta \sum_{l \in \mathcal{N}_k/\{k\}} b_{kl}\|f - f_l^o\|_{\mathcal{F}}^2\,, \tag{9.41}$$

for some $\{b_{kl}\}$ such that $\sum_{l\in\mathcal{N}_k} b_{kl} = 1 \;\; \forall\; k$.

By now using the steepest decent algorithm in (9.20), and denote the approximation of f at time n and at node k by ψ_k^n, we get:

$$\begin{aligned}
\psi_k^n &= \psi_k^{n-1} - \frac{\mu_{n,k}}{2}\nabla_f J_k^{\mathcal{N}}(\psi_k^{n-1}) \\
&= \psi_k^{n-1} - \frac{\mu_{n,k}}{2}\nabla_f \left(J_k(\psi_k^{n-1}) + \delta \sum_{l \in \mathcal{N}_k/\{k\}} b_{kl}\|\psi_k^{n-1} - f_l^o\|_{\mathcal{F}}^2 \right) \\
&= \psi_k^{n-1} - \frac{\mu_{n,k}}{2}\nabla_f J_k(\psi_k^{n-1}) - \frac{\mu_{n,k}}{2}\delta \sum_{l \in \mathcal{N}_k/\{k\}} b_{kl}\nabla_f\|\psi_k^{n-1} - f_l^o\|_{\mathcal{F}}^2 \\
&= \psi_k^{n-1} - \frac{\mu_{n,k}}{2}\nabla_f J_k(\psi_k^{n-1}) - \mu_{n,k}\delta \sum_{l \in \mathcal{N}_k/\{k\}} b_{kl}(\psi_k^{n-1} - f_l^o).
\end{aligned}$$

This can be split into a two step process

$$\begin{aligned}\phi_k^n &\leftarrow \psi_k^{n-1} - \frac{\mu_{n,k}}{2}\nabla_f J_k(\psi_k^{n-1}),\\ \psi_k^n &\leftarrow \phi_k^n - \mu_{n,k}\delta \sum_{l\in\mathcal{N}_k/\{k\}} b_{kl}\left(\psi_k^{n-1} - f_l^o\right).\end{aligned} \tag{9.42}$$

By now substituting ϕ_k^n for ψ_k^{n-1} and using the local estimate, ϕ_l^n, as an approximation for the local optimal, f_l^o, the right hand side of the second equation in (9.42) takes the form:

$$\begin{aligned}\phi_k^n - \mu_{n,k}\delta \sum_{l\in\mathcal{N}_k/\{k\}} b_{kl}(\phi_k^n - \phi_l^n) &= (1-\mu_{n,k}\delta+\mu_{n,k}\delta b_{kk})\phi_k^n + \sum_{l\in\mathcal{N}_k/\{k\}} \mu_{n,k}\delta b_{kl}\phi_l^n\\ &= \sum_{l\in\mathcal{N}_k} a_{kl}\phi_l^n \quad \text{with} \quad a_{kl} = \begin{cases} 1-\mu_{n,k}\delta+\mu_{n,k}\delta b_{kk} & \text{if } k=l\\ \mu_{n,k}\delta b_{kl} & \text{otherwise.}\end{cases}\end{aligned}$$

Note that since $\sum_{l\in\mathcal{N}_k} b_{kl} = 1$ we have $\sum_{l\in\mathcal{N}_k} a_{kl} = 1$ for all k. If we choose $\delta = 1/\mu_{n,k}$ we furthermore get $a_{kl} = b_{kl}$. Using the above derived expressions we get the new form of (9.42) as

$$\begin{aligned}\phi_k^n &\leftarrow \psi_k^{n-1} - \frac{\mu_{n,k}}{2}\nabla_f J_k(\psi_k^{n-1}),\\ \psi_k^n &\leftarrow \sum_{l\in\mathcal{N}_k} a_{kl}\phi_l^n.\end{aligned} \tag{9.43}$$

We have according to (9.24) that $\nabla_f J_k(f) = H_k(f)$. This transforms algorithm (9.43) into (9.40). Thus we have shown how the ATC method can be motivated. The motivation for the CTA method is very similar only switching the order that some of the steps are done in altering how the approximations are done.

Combination Rules for Diffusion Methods

There are several ways to select the combination weights $\{a_{kl}\}$ for the diffusion methods in (9.39) and (9.40). Here we will discuss some of the most common variants. Some of these are very simple other more complicated, especially the adaptive ones that will be considered last. We will here use the notation of $A = \{a_{kl}\}$ for a $(K \times K)$ matrix containing all the combination weights. The combination weights are conformed to the following:

$$a_{kl} \geq 0, \quad \sum_{l=1}^{K} a_{kl} = 1, \quad \text{and} \quad a_{kl} = 0 \quad \text{if} \quad l \notin \mathcal{N}_k, \tag{9.44}$$

which implies that

$$A\mathbb{1} = \mathbb{1}, \tag{9.45}$$

where we have $\mathbb{1}$ according to (9.35), making A a right-stochastic matrix. These are the same requirements that were put on the matrix of coefficients $C = \{c_{kl}\}$ found in (9.33) and (9.34). This means that all the following choices for $\{a_{kl}\}$ will work for $\{c_{kl}\}$ as well.

The simplest way of deciding the weights is just to choose them so to average the incoming values, with

$$n_k \triangleq |\mathcal{N}_k| = \text{number of nodes in the neighborhood including } k \text{ itself}$$

this means

$$a_{kl} = \frac{1}{n_k} \qquad \forall \;\; l \in \mathcal{N}_k \, .$$

This is a very robust choice which exploits the connectivity of the network quite fully. It is also easily adapted to node failures and network topology changes as the weights only depend on the number of current connections.

Consider the Laplacian matrix below:

$$\mathcal{L}_{kl} = \begin{cases} -1 & \text{if } k \neq l \text{ are linked} \\ n_k - 1 & \text{if } k = l \\ 0 & \text{otherwise} \end{cases}$$

The weights are the chosen as follows:

$$A = I_N - \gamma \mathcal{L}_{kl} \, ,$$

for some constant γ. By choosing $\gamma = n_{\max}$ where $n_{\max}$ is the maximum degree across the network we end up with

$$a_{kl} = \begin{cases} 1/n_{\max} & \text{if } k \neq l \text{ are linked} \\ 1 - (n_k - 1)/n_{\max} & \text{if } k = l \\ 0 & \text{otherwise} \end{cases} \, . \tag{9.46}$$

Another choice is just $\gamma = K$, the size of the network, giving

$$a_{kl} = \begin{cases} 1/K & \text{if } k \neq l \text{ are linked} \\ 1 - (n_k - 1)/K & \text{if } k = l \\ 0 & \text{otherwise} \end{cases} \, . \tag{9.47}$$

The Metropolis rule considers how connected the neighbors are as well, with n_k and n_l denoting the degree of node k and l respectively, we have

$$a_{kl} = \begin{cases} 1/\max(n_k, n_l) & \text{if } k \neq l \text{ are linked} \\ 1 - \sum_{l \in \mathcal{N}_k / \{k\}} a_{kl} & \text{if } k = l \\ 0 & \text{otherwise} \end{cases} \, . \tag{9.48}$$

A similar choice is the following, that does not yield symmetric weights, which the Metropolis rule does, but generally performs better is

$$a_{kl} = \begin{cases} n_l / \sum_{m \in \mathcal{N}_k} n_m & \text{if } k \text{ and } l \text{ are linked or } k = l \\ 0 & \text{otherwise} \end{cases}. \tag{9.49}$$

The above described weights are, as long as the network topology remains intact, constants in time. This is of course not the case for all combination weights. Here below we we consider the so called adaptive combination weights. The idea here is to consider the relative measurement noise levels at the nodes and weight them thereafter, giving the nodes with lower noise higher weights. The two most commonly used combination rules are Hastings rule and the relative-variance rule. The Hastings rule is described in the following equation, where $\sigma^2_{v,l}$ is the measurement variance at node l.

$$a_{kl} = \begin{cases} \dfrac{\sigma^2_{v,l}}{\max\{n_k \sigma^2_{v,k}, n_l \sigma^2_{v,l}\}} & \text{if } k \neq l \text{ are linked} \\ 1 - \sum_{l \in \mathcal{N}_k / \{k\}} a_{kl} & \text{if } k = l \\ 0 & \text{otherwise.} \end{cases} \tag{9.50}$$

The relative-variance rule on the other hand is:

$$a_{kl} = \begin{cases} \dfrac{\sigma^{-2}_{v,l}}{\sum_{l \in \mathcal{N}_k} \sigma^{-2}_{v,l}} & \text{if } k \text{ and } l \text{ are linked or } k = l \\ 0 & \text{otherwise.} \end{cases} \tag{9.51}$$

There is one major problem with (9.50) and (9.51) is that we in general know nothing about the variances $\sigma^2_{v,l}$. This means that to be able to use the mentioned combination rules we have to have a way of approximating these values.

Working towards this goal we first consider the following filters:

$$\gamma^2_{kl}(n) = (1 - \nu_k)\gamma^2_{kl}(n-1) + \nu_k \|\phi^{n-1}_l - \phi^{n-1}_k\|^2, \quad \text{(CTA)} \tag{9.52}$$

$$\zeta^2_{kl}(n) = (1 - \nu_k)\zeta^2_{kl}(n-1) + \nu_k \|\psi^{n}_l - \phi^{n-1}_k\|^2, \quad \text{(ATC)} \tag{9.53}$$

where $\nu_k \in (0, 1)$ is a small positive coefficient. The reason that we have two of them is that they use the combination weights at different times in each iteration. In the ATC version we have to save the value of ϕ^{n-1}_k since this is the latest value that is not influenced by the measurements from neighboring nodes in general. Remember that the adaptive state may incorporate information from its neighborhood. It can be verified that as the number of iterations goes towards infinity, i.e., $n \to \infty$, the values of γ^2_{kl} and ζ^2_{kl} approximately approaches a constant multiplier times the sought $\sigma^2_{v,l}$.

Since this constant multiplier is common to all γ_{kl}^2 and ζ_{kl}^2 we can use them directly in the place for $\sigma_{v,l}^2$ in both (9.50) and (9.51). Notably this allows us to get adaptive combination weights without sending any more information than we already do. Some of the above discussed combination rules does however require us to send additional information in form of the node degrees. This increases the required communication, but we can for example only update these numbers every tenth iteration or suchlike, assuming that the topology does not change considerably in each iteration. If the topology does not change at all, we can just perform this communication step once and hence the problem vanishes.

9.2.3 Distributed Learning Using Kernel Methods

In this section we are going to discuss how we can use kernel methods in distributed learning in WSNs. In a previous section we already handled the case with a star topology network and so we will now consider a general network. In the kernel methods we use so called training data to find the desired learning rule. This training data can be assumed to have been acquired before the communication begins or in steps during the algorithm. The training data in WSNs is usually pairs of $(x_k, d_k(n))$, where x_k usually is the position of the node which may or may not include the time and d_k the corresponding measurement of y.

We are now going to consider how we can apply the derived learning algorithms from former sections for the specific case of kernel methods. This means that we are searching for a function $f \in \mathcal{H}_K$ that for a x gives us a y, where x could be the space-time position and y the quantity that we are interested in, e.g. the temperature. Since we are interested in the function that predicts y which we measure directly, in $d_{n,k}$, we get the simple $h_k(f) = f(x_k)$ in (9.26) giving

$$J_k^{\text{loc}}(f) = \mathbb{E}\left\{\|f(x_k) - d_k\|^2\right\}. \tag{9.54}$$

Because $f \in \mathcal{H}_K$ we have that $h_k(f) = f(x_k) = \langle f, \kappa_{x_k}\rangle_{\mathcal{H}_K}$, which in (9.28) gives us:

$$\begin{aligned} H_k^{\text{loc}}(f) = \nabla_f J_k^{\text{loc}}(f) &= \mathbb{E}\left\{2(\langle f, \kappa_{x_k}\rangle_{\mathcal{H}_K} - d_k)\frac{d\langle f, \kappa_{x_k}\rangle_{\mathcal{H}_K}}{df}\right\} \\ &= \mathbb{E}\left\{2(\langle f, \kappa_{x_k}\rangle_{\mathcal{H}_K} - d_k)\kappa_{x_k}\right\} = \mathbb{E}\left\{2(f(x_k) - d_k)\kappa_{x_k}\right\}. \end{aligned}$$

Since we do not know the statistical distributions on the variables in the equation above we need some way to approximate the expectation value. Without any prior knowledge we use the instantaneous estimate $\mathbb{E}\{X\} = X$, i.e., LMS approximation, giving us

$$H_k^{\text{loc}}(f) = 2(f(x_k) - d_k)\kappa_{x_k}. \tag{9.55}$$

With the finished expression for $H_k(f)$ in (9.55) we can use the previously derived framework for the cyclic path and diffusion methods in (9.30), (9.39) and (9.40).

Cyclic Path Method

The cyclic path method in (9.30) uses the $H_k^{\text{loc}}(f)$ in each update. Here however we would like to generalize this a little bit by lending the complexity controlling term in (9.37). This is done since we are optimizing a function which we would like to stay relatively uncomplicated. This causes no problems in the algorithm since this term, given the set $\{\lambda_k\}_{k=1}^K$, will not need any other information not already pertained in the original algorithm. This will give us

$$H_k(f) = 2(f(x_k) - d_k)\kappa_{x_k} + 2\lambda_k f.$$

If this is put into (9.30) instead of H_k^{loc} we get, where we have put a time index n on x_k to indicate that they can be time dependent:

$$\begin{aligned}
&\psi_0^n \leftarrow f_{n-1}, \\
&\psi_k^n \leftarrow \psi_{k-1}^n(1 - \mu_{k,n}\lambda_k) - \mu_{k,n}(\psi_{k-1}^n(x_{k,n}) - d_k(n))\kappa_{x_{k,n}}, \qquad k = 1, \ \dots \ , K, \\
&f_n \leftarrow \psi_K^n.
\end{aligned} \tag{9.56}$$

This algorithm requires the nodes to send and receive functions, which might sound very abstract and communication inefficient. This is certainly true for the general case, but we have restricted our function to $\mathcal{H}_K$ and if we preprogram the sensors with the kernel they only need to know the input values $\{x_i\}_{i=1}^m$ and the corresponding weights c_m^λ as in (9.8). If the input at the nodes does not change, these vectors will be bounded to the size of the network. If the network is big however this could still be much data. This set of data is exactly the same size as the total training data sat and thus, we could as well just let the sensors relay this information to a fusion center which could use the technique in (9.9), which probably gives better results. Thus this method might not be to useful.

Diffusion Methods

Using the general derived $H_k^{\text{gen}}(f)$ in (9.37) and the the $H_k^{\text{loc}}(f)$ in (9.55) we get:

$$H_k^{\text{gen}}(f) = 2\sum_{l \in \mathcal{N}_k} c_{lk}(f(x_l) - d_l)\kappa_{x_l} + 2\lambda_k f, \tag{9.57}$$

which in gives us the diffusion methods in (9.39) and (9.40) as:

General CTA diffusion LMS

$$\phi_k^n \leftarrow \sum_{l \in \mathcal{N}_k} a_{kl} \psi_l^{n-1},$$
$$\psi_k^n \leftarrow \phi_k^n (1 - \mu_{n,k}\lambda_k) + \mu_{n,k} \sum_{l \in \mathcal{N}_k} c_{lk}(d_l(n) - \phi_l^n(x_{l,n}))\kappa_{x_{l,n}}. \tag{9.58}$$

General ATC diffusion LMS

$$\phi_k^n \leftarrow \psi_k^{n-1}(1 - \mu_{n,k}\lambda_k) + \mu_{n,k} \sum_{l \in \mathcal{N}_k} c_{lk}(d_l(n) - \psi_l^{n-1}(x_{l,n}))\kappa_{x_{l,n}},$$
$$\psi_k^n \leftarrow \sum_{l \in \mathcal{N}_k} a_{kl} \phi_l^n. \tag{9.59}$$

These algorithms generally perform very well. The problem is again that we are passing functions between nodes, this time to all neighboring nodes. This requires extensive communication and might hence only be plausible to implement where the energy consumption requirements are not too limiting. If we assume that the x_l do not vary with time, the communicated vectors representing the functions will be bounded and we could save some data compared to the case where all information is relayed to a fusion center. The real gain with this method however is that it, in comparison to both centralized solutions and the cyclic path method, is far more robust. Here we have no crucially important nodes and if the combination rules can handle node failures the system will work just fine if a couple of nodes fail, whereas the other methods might or will fail.

Alternating Projection Algorithms

Here we address the issue of the above described methods. We thereby seek a distributed method for finding a decision rule that does not involve sending functions but rather real valued numbers. To construct the algorithm we assume that each sensor gets one training data sample (x_k, y_k). The algorithm could be constructed more generally by letting number of data samples be bigger. Here however we mostly seek to convey the idea of the algorithm. Further assume that sensor k can query the data (x_l, y_l) from its neighbors $l \in \mathcal{N}_k$. Then each sensor could solve the local approximation of the global problem in (9.5) using the squared error loss function by only considering its neighborhood:

$$\min_{f \in \mathcal{H}_K} \left[\sum_{l \in \mathcal{N}_k} (f(x_l) - y_l)^2 + \lambda_k \|f\|_{\mathcal{H}_K}^2 \right]. \tag{9.60}$$

The solution to this problem can be solved locally and all sensors would have a local estimate. This however does not use the entire networks data

and resemble more a network of sensors taking turns at being fusion center. The idea of the alternating projection algorithm is to include a set of message variables $\{z_l\} \in \mathbb{R}$, one for each sensor. These can then be read and changed by all sensors in $\mathcal{N}_k$. Each sensor initializes with $z_l = y_l$ then the sensors take turns in querying its neighbors message variables and solving its neighborhood problem. When this is done it updates its neighbors variables according to the function it just found. When a cycle is done the same procedure is repeated until some convergence criteria is reached. We call the estimate at sensor k at time n for f_k^n. A little change is made to the problem above so that sensor k controls the inertia of the algorithm by changing the complexity term of (9.60). This makes the problem that sensor k solves at iteration n:

$$\min_{f \in \mathcal{H}_K} \left[\sum_{l \in \mathcal{N}_k} (f(x_l) - z_l)^2 + \lambda_k \|f - f_k^n\|_{\mathcal{H}_K}^2 \right] . \tag{9.61}$$

The resulting algorithm can be seen in Table 9.2 and is also depicted in Figure 9.15. Here this algorithm has only been motivated by intuitive arguments, however the algorithm can be analyzed formally by using successive orthogonal projection, SOP, algorithms applied to network typologies and kernel methods. This is however far outside the scope of this book and will hence be omitted, the analysis can be found in (Predd et al., 2005) and (Predd et al., 2006b) for the interested reader. By using the framework for SOP algorithms it can be shown that the algorithm in Table 9.2 converges in the limit $n \to \infty$ and does so to an approximation of the global optimum. That it does not the optimal solution can be understood by considering that it solves a optimization problem of lower degree at each node than would be done centrally and thus the solution according to the Representer theorem in (9.8) is of lower degree. Thus we can only reach the best possible approximation that the sub-dimensional space offers for the real solution.

A few comments on the algorithm above are in order. We have achieved an algorithm that only sends, in the iteration, real valued numbers and hence is much more suited for many situations where the communication constraints are strong. As stated above in (9.61), each sensor has to solve an optimization problem, which might seem abstract. This can however be done in a similar manner as to (9.9), which involves solving a $|\mathcal{N}_k|$-dimensional system of linear equations. Another thing worth mentioning is that as stated now the the algorithm in Table 9.2 the sensors perform their local calculations in order. The calculations could be parallelized as long as none of the message variables z_l are updated simultaneously. We mentioned, in the beginning of this section, that the assumption that each sensor only contribute with one training data sample could be loosened to include many samples. If this is the case the sum of squared errors would include all the training data samples of the neighboring sensors.

Table 9.2 Training distributively with alternating projections

Initialization:	Neighboring sensors share training data inputs: sensor k stores $\{x_l\}_{l\in\mathcal{N}_k}$ Sensor k initializes $z_k = y_k$, $f_k^n = 0 \in \mathcal{H}_K$
Training:	for $n = 1, ..., N$ for $k = 1, ..., K$ Sensor k: Queries $z_l \quad \forall\, l \in \mathcal{N}_k$ $f_k^n := \arg\min_{f\in\mathcal{H}_K} \Big[\sum_{l\in\mathcal{N}_k}(f(x_l) - z_l)^2 + \lambda_k \|f - f_k^{n-1}\|^2_{\mathcal{H}_K}\Big]$ Updates $z_l \leftarrow f_k^n(x_l) \quad \forall\, l \in \mathcal{N}_k$ end end

9.2.4 Distributed Learning Using ARMA-time Series

In this section we assume that the phenomena we are interested in learning about can be described by ARMA-time series according to

$$d_k(n) = u_k(n)\omega^o + v_k(n), \tag{9.62}$$

where $d_k(n)$ is the scalar parameter we are interested in predicting, $u_k(n)$ the history of the system at node k containing previous system states and interference, $v(n)$ is some white noise and ω^o is the vector that describes how the system works, see Section 9.1.2. We use M as the length of u_k and ω^o as in Section 9.1.2.

The goal of the network is to find ω^o, which is defined as

$$\omega^o = \arg\min_{\omega} \sum_{k=1}^{K} J_k(\omega).$$

Because of this we use ω^o to denote the optimal solution and ω to denote an estimate of this parameter. Because of the structure in (9.62) we assume that the optimal solution ω^o also is the optimal solution to the local cost functions. This is the case for the cost functions that we are going to use.

We assume that the data $\{d_k(n), u_{k,n}\}$ follows the following criteria:

1. The unknown vector ω^o relates the data $\{d_k(n), u_{k,n}\}$ as in (9.62) where $v_k(n)$ is some white noise sequence with variance $\mathbb{E}\{\|v_k(n)\|^2\} = \sigma^2_{v,k}$ which is independent of $\{d_k(n), u_{k,n}\}$ for all k, n.

2. The sequences $\{u_{k,n}\}$ are spatially independent, i.e., $u_{k,n}$ is independent of $u_l(n)$ for $k \neq l$.

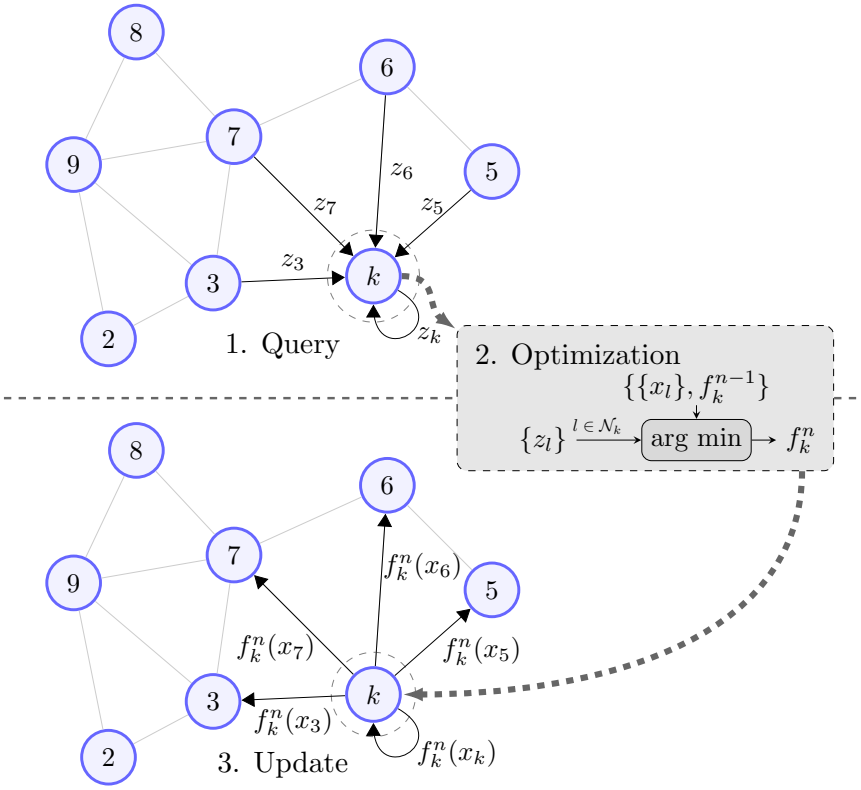


Figure 9.15 The alternating projection algorithm in use. All nodes perform three ordered steps in each time step: Querying, Optimization and Updating. The optimization step finds the optimal function according to (9.61).

3. The sequence $\{u_{k,n}\}$ is independent over time, i.e., time independent.

4. The regressors $\{u_{k,n}\}$ arise from a temporally white distribution.

Some of these criteria might not be completely fulfilled in many situations, especially when employing a MA-part the time independence part is a bit problematic. Here the regressors exhibit a time shift structure. However, there have been extensive studies in stochastic approximation literature showing that results based on the time independent approximation still match well with the actual solutions for sufficient small step sizes. Therefore we will use the above conditions and assume that sufficiently small step sizes are used.

It is worth mentioning what capabilities the nodes are assumed to have. The general ARMA-time series contains both an auto-regressive, AR, and a moving-average, MA, part. The AR part is straight forward, it only requires the nodes to store a certain number of old measurements. The MA part however assumes that we know how the past disturbances to the system

have looked. This translates to that we either can measure these disturbing forces or that the node itself produces them, and know their appearance. Thus it requires the nodes to have sensors that can measure the disturbances or probing equipment to produce the wanted disturbances. Because of this simpler sensors might only be capable of implementing the AR part. This is not too limiting however since some unknown disturbances could be factored into the white noise term $v_k(n)$ in (9.62). From here on out we will not distinguish between ARMA-time series containing an AR part, a MA part or both since the analysis is the same regardless.

We denote the $M \times M$ covariance matrices of the data u by

$$R_{u,k} = \mathbb{E}\{u_k^T u_k\}, \tag{9.63}$$

and the $M \times 1$ cross-covariance vectors by

$$R_{du,k} = \mathbb{E}\{d_k u_k^T\}. \tag{9.64}$$

Given this model we see that we are trying to predict $d(n)$ according to (9.62) and hence we get, skipping the index for the time n, $h_k(\omega) = u_k\omega$. This can be used to calculate different $H_k(f)$ that enable the use of the previously derived algorithms in (9.30), (9.39) and (9.40) to find the desired ω^o.

Cyclic Methods

In the cyclic method we use the local MSE cost function and $h_k(\omega) = u_k\omega$ in (9.28) gives us:

$$\begin{aligned} H_k^{\text{loc}}(\omega) &= \mathbb{E}\left\{2(u_k\omega - d_k)\frac{d\ u_k\omega}{d\omega}\right\} = \mathbb{E}\left\{2(u_k\omega - d_k)u_k^T\right\} \\ &= 2\,\mathbb{E}\left\{u_k^T u_k\omega - d_k(n)u_k^T\right\} = 2(R_{u,k}\omega - R_{du,k}), \end{aligned}$$

so we have

$$H_k^{\text{loc}}(\omega) = 2(R_{u,k}\omega - R_{du,k}), \tag{9.65}$$

with $R_{u,k}$ and $R_{du,k}$ according to (9.63) and (9.64). Inserting (9.65) into (9.30) then gives:

$$\begin{aligned} &\psi_0^n \leftarrow \omega_{n-1}, \\ &\psi_k^n \leftarrow \psi_{k-1}^n - \mu_k(R_{u,k}\psi_{k-1}^n - R_{du,k}), \qquad k = 1,\ \ldots\ , K, \\ &\omega_n \leftarrow \psi_K^n. \end{aligned} \tag{9.66}$$

Here we have assumed temporally homogeneous step sizes but allowed spatial differences.

However since we generally do not know the statistical profile, and hence $\{R_{u,k}, R_{du,k}\}$, we must approximate these to be able to use the algorithm in

general. This is done by using local instantaneous approximations according to the LMS type, i.e.,

$$R_{u,k} \approx u_{k,n}^T u_{k,n}, \qquad R_{du,k} \approx d_k(n) u_{k,n}^T. \tag{9.67}$$

This gives a new much more applicable algorithm according to:

Cyclic Incremental LMS

$$\begin{aligned} &\psi_0^n \leftarrow \omega_{n-1}, \\ &\psi_k^n \leftarrow \psi_{k-1}^n + \mu_k u_{k,n}^T (d_k(n) - u_{k,n}\psi_k^n), \qquad k = 1, \ \dots, K, \\ &\omega_n \leftarrow \psi_K^n. \end{aligned} \tag{9.68}$$

Here one could initiate with e.g. $\omega_{-1} = 0$ and repeat the algorithm above until good enough precision is achieved.

Diffusion Methods

Now when considering diffusion methods we are allowed much greater freedom in the choice of cost functions $J_k(\omega)$. We start by using the general cost functions described in (9.36). This gives us a $H_k(\omega)$ according to (9.37):

$$H_k^{\text{gen}}(\omega) = \sum_{l \in \mathcal{N}_k} c_{lk} H_l^{\text{loc}}(\omega) + \lambda_k \omega. \tag{9.69}$$

Since we are searching for a vector ω_o, and not a function, we do not need to incorporate the λ-term which regularizes the algorithm not to generate to big and complex answers. The desired ω^o might be a vector of large norm, but that is the vector we want to find and if the model is correct it should not cause any bigger problems. This motivates us to simplify (9.69):

$$H_k^{\text{gen}}(\omega) = \sum_{l \in \mathcal{N}_k} c_{lk} H_l^{\text{loc}}(\omega).$$

Now using (9.65) and the LMS approximations in (9.67), omitting the time indicator n, we instead get

$$H_k^{\text{gen}}(\omega) = 2 \sum_{l \in \mathcal{N}_k} c_{lk} u_k^T (u_k \omega - d_k). \tag{9.70}$$

By using the derived $H_k(\omega)$ from (9.70) in the algorithm for the diffusion methods, (9.39) and (9.40), we get:

General CTA diffusion LMS

$$\begin{aligned} &\phi_k^n \leftarrow \sum_{l \in \mathcal{N}_k} a_{kl} \psi_l^{n-1}, \\ &\psi_k^n \leftarrow \phi_k^n + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk} u_{l,n}^T (d_l(n) - u_{l,n}\phi_l^n). \end{aligned} \tag{9.71}$$

General ATC diffusion LSM

$$\phi_k^n \leftarrow \psi_k^{n-1} + \mu_k \sum_{l \in \mathcal{N}_k} c_{lk} u_{l,n}^T (d_l(n) - u_{l,n}\psi_l^{n-1}),$$
$$\psi_k^n \leftarrow \sum_{l \in \mathcal{N}_k} a_{kl}\phi_l^n. \tag{9.72}$$

When using these algorithms one usually initialize with $\{\psi_l^{-1} = 0\}$ for all l and then iterate the algorithms above once for each node in each time step $n \geq 0$. In both of these algorithms all sensors need send information two times in each iteration, one time in the combine step where they send their temporary local estimate and another time in the adapt step where they share their local correction term, $u_{k,n}^T(d_k(n) - u_{k,n}\psi_k^{n-1})$. Both times the conveyed message is a vector of length M. This makes the algorithm quite communication heavy and a bit slower than it has to be. The algorithms in (9.71) and (9.72) are very versatile and robust but might in some situations be a bit to complicated. In many cases we would like to lower the communication requirements and thus we discuss a simpler version of the above. By only using the local data in the adapt step, i.e., $c_{lk} = \delta_{lk}$ (δ_{lk} is as before the Kronecker delta) or equivalently using $J_k(\omega) = J_k^{\text{loc}}(\omega)$, we can simplify algorithms (9.71) and (9.72) to

Local CTA diffusion LMS

$$\phi_k^n \leftarrow \sum_{l \in \mathcal{N}_k} a_{kl}\psi_l^{n-1},$$
$$\psi_k^n \leftarrow \phi_k^n + \mu_k u_{l,n}^T (d_l(n) - u_{l,n}\phi_l^n). \tag{9.73}$$

Local ATC diffusion LSM

$$\phi_k^n \leftarrow \psi_k^{n-1} + \mu_k u_{l,n}^T (d_l(n) - u_{l,n}\psi_l^{n-1}),$$
$$\psi_k^n \leftarrow \sum_{l \in \mathcal{N}_k} a_{kl}\phi_l^n. \tag{9.74}$$

Here we can as earlier proposed initiate with $\phi_l^{-1} = 0$ for all l and then iterate the above described algorithm for $n \geq 0$ until good enough convergence is reached. Here we only use local data for the adaptive step and thus Figure 9.13 and 9.14 show exactly how this method works. The above described methods are often a very useful in practice and not to communication demanding or complicated. We shall in the next chapter discuss how the discussed methods perform concerning convergence speed and precision.

9.2.5 Convergence Speed and Precision

In this section we will look at the convergence of some of the described algorithms. Special focus will be given methods using ARMA-time series

under certain constraints. This is because of their simpler form to the kernel methods making them less troublesome to analyze. We will however try to communicate how these results can give clues to how the given conclusions can be thought of in general.

First of all let us decide what restrictions we will apply when performing the analysis. We will restrict our attention to the case of ARMA-methods and use the criteria on the data presented in the numbered list in the beginning of section 9.2.4. We will further assume , i.spatially independent regressor covariancese. $R_{u,k} = R_u$ for all k where $R_{u,k}$ is defined as in (9.63). Additionally we restrict our cast to $R_u > 0$, i.e., that R_u is positive-definite, and hence invertible. We further assume that the step sizes used in the algorithms are of the same size, i.e., $\mu_{n,k} = \mu$. We will regardless of algorithm assume that we use the local MSE cost function, i.e., we have $J_k(\cdot) = J_k^{\text{loc}}(\cdot)$ where $J_k^{\text{loc}}(\cdot)$ is as in (9.26). This means that we limit us to the algorithms in (9.66), (9.73) and (9.74). If we assume all the criteria described in this paragraph are fulfilled all the results and theorems that following analysis yields are correct.

The Non-cooperative Case

First, before looking at the distributed cases we discuss the case where the nodes do not cooperate. This is important to see how the cooperative algorithms perform in comparison. We will be considering the mean square deviation, MSD, as time goes towards infinity and we define:

$$\text{MSD}_k = \lim_{t\to\infty} \mathbb{E}\left\{\|\omega^o - \omega_{k,t}\|^2\right\}, \tag{9.75}$$

where $\omega_{k,t}$ is the estimate at node k by the time t. The time limits can be translated to a limits in n the number of steps the algorithm in question has taken.

Let us start with the case where each sensor uses the steepest decent method and the local MSE cost function with the LMS approximation. Here wee do not allow any cooperation. In this case we get:

$$\omega_k^{n+1} = \omega_k^n + \mu u_{k,n}^T (d_k(n) - u_{k,n}\omega_k^n).$$

It is known that for sufficiently small μ this will converge and we get

$$\text{MSD}_{\text{ncop},k} \approx \frac{\mu M}{4}\sigma_{v,k}^2, \tag{9.76}$$

where M is the length of ω and the ncop in the subscript indicates that this is for the non-cooperative case. It is further known that the condition for convergence is that μ obeys:

$$\mu < \frac{4}{\lambda_{\max}(R_u)}, \tag{9.77}$$

and for such μ the MSD has a convergence rate, towards its steady state value, of

$$r \approx 1 - \mu \cdot \lambda_{\min}(R_u)\,, \tag{9.78}$$

where $\lambda_{\max}(R_u)$ and $\lambda_{\min}(R_u)$ are the biggest and smallest eigenvalues of R_u respectively. The smaller $r \in (0,1)$ the faster the convergence. We will not prove either of these but rather simply consider them as true.

By averaging the MSD_k over the network we get the network MSD as:

$$\text{MSD}_{\text{ncop}}^{\text{network}} \approx \frac{\mu M}{4} \cdot \left(\frac{1}{K} \sum_{l=1}^{K} \sigma_{v,l}^2 \right). \tag{9.79}$$

The Centralized Case

Now we consider the case where all the information is sent to a fusion center as in the star topology case. In this case the fusion center can use an average of the local cost functions as the global cost function giving the algorithm

$$\omega_{n+1} = \omega_n + \mu \left(\frac{1}{K} \sum_{l=1}^{K} u_{l,n}^T (d_l(n) - u_{l,n}\omega_n) \right).$$

This algorithm will have a better MSD according to

$$\text{MSD}_{\text{cent}} \approx \frac{\mu M}{4} \cdot \frac{1}{K} \cdot \left(\frac{1}{K} \sum_{l=1}^{K} \sigma_{v,l}^2 \right), \tag{9.80}$$

i.e., a K-fold performance improvement. Note that this is without degrading the convergence rate which remains the same as in (9.78). We also have the same condition (9.77) for convergence. Now the question is however the distributed algorithms can perform equally well. This question is attended to in the following sections.

The Cyclic Algorithm

Now we turn our attention to the incremental cyclic algorithm in (9.68). If we take some inspiration from the centralized case and introduce a factor of $1/K$ to cycle each step, so to make the step size for each time step n equal to μ and not $K \cdot \mu$. We then get

$$\text{MSD}_{\text{cycl}} \approx \text{MSD}_{\text{cent}} \approx \frac{\mu M}{4} \cdot \frac{1}{K} \cdot \left(\frac{1}{K} \sum_{l=1}^{K} \sigma_{v,l}^2 \right), \tag{9.81}$$

while still keeping (9.77) and (9.78), as in the non-cooperative case. It is worth noting however that the convergence rate is in time steps n and how

long time one step corresponds to is different from algorithm to algorithm. Here in the cyclic path one time step includes a whole cycle through the network, which most certainly will take longer time than processing everything centrally, since there is a lot of idle waiting time involved with the cyclic algorithm.

Diffusion methods

When we consider the Diffusion methods, the analysis is more complex. Remember that we have restricted our attention to the local CTA and ATC LMS methods in (9.73) and (9.74). The analysis is complicated and involves advanced linear algebra. Thus we will only present the following facts and theorems.

First, we can conclude that the convergence rate for diffusion networks is still given by (9.78) and that the condition (9.77) still holds. As we mentioned before there are performance differences between the CTA and the ATC algorithms. The convergence rates are both the same as stated above. The MSD is not however as stated by the following theorem (9.2.6). Remember that the matrix A is the matrix containing all the combination weights for the network.

Theorem 9.2.6 (**Comparing MSD Performance**). *Assume that A is symmetric or close to symmetric. Then the ACT diffusion method achieves the lowest network MSD and*

$$\mathrm{MSD}_{\mathrm{ATC}}^{\mathrm{network}} \leq \mathrm{MSD}_{\mathrm{CTA}}^{\mathrm{network}} \leq \mathrm{MSD}_{\mathrm{ncop}}^{\mathrm{network}} . \tag{9.82}$$

Now we are interested in how the MSD performance is at the individual nodes. The restrictions on the combination weights $\{a_{kl}\}$ found in (9.44) and the assumption that the network is connected gives the matrix $A = \{a_{kl}\}$ a new property. A is then what is known as a primitive matrix, which in turn implies that A has a unique eigenvalue at one and that all other eigenvalues will have magnitude less than one. If we denote the eigenvector corresponding to the eigenvalue at one with p which we have normalized so that its terms add up to one. $p = \mathrm{col}\{p_1, p_2, ..., p_K\}$ will then be defined by:

$$Ap = p, \quad p^T \mathbb{1} = 1, \quad 0 < p_l < 1 , \tag{9.83}$$

where we have $\mathbb{1}$ according to (9.35). With this definition we now have the next theorem.

Theorem 9.2.7 (**Node and Network Diffusion MSD**). *For connected diffusion networks the performance of each individual node is approximately*

equal to the network MSD and they are both well approximated by

$$\text{MSD}_{\text{diff,k}} \approx \text{MSD}_{\text{diff}}^{\text{network}} \approx \frac{\mu M}{4}\left(\sum_{l=1}^{K} p_l^2 \sigma_{v,l}^2\right) + \mathcal{O}(\mu^2) \tag{9.84}$$

for all nodes k.

Comparing (9.84)to (9.80) in the centralized case we have that the effect of diffusion cooperation is to scale the noise variances by the factor $\{p_l\}$ instead of the previous $1/K^2$. These factors are determined by the combination rule matrix A. Note that Theorem 9.2.7 does not limit us to either CTA or ATC which means that the performance difference found in Theorem 9.2.6 must arise from second order terms in μ.

To get a better picture of what the factors $\{p_l\}$ means we look at the case where A is symmetric. This is the case for some of the previously discussed combination rules. Anyhow a symmetric A gives us

$$A^T \mathbb{1} = A\mathbb{1} = \mathbb{1}\,, \tag{9.85}$$

which in turn implies that $p = \mathbb{1}/K$ or $p_l = 1/K$ for all nodes l. Thus equation (9.84) takes the form, where we have neglected the second order terms:

$$\text{MSD}_{\text{diff,sym,k}} \approx \text{MSD}_{\text{diff,sym}}^{\text{network}} \approx \frac{\mu M}{4} \cdot \frac{1}{K} \cdot \left(\frac{1}{K}\sum_{l=1}^{K} \sigma_{v,l}^2\right), \tag{9.86}$$

where sym in the subscript indicates that A is symmetric. This is the same MSD as for the centralized case. Lastly we want to examine however using adaptive combination rules enhances performance. To achieve the best performance we want to solve the following problem:

$$A^o = \arg\min_{A \in \mathcal{A}} \sum_{l=1}^{K} p_l \sigma_{v,l}^2\,,$$

where $\mathcal{A}$ denote all the $(K \times K)$ matrices that satisfy (9.44). Interestingly it turns out that we have already discussed one solution to the equation above. The so called Hastings rule in (9.50) is a solution. The resulting minimum (optimal) MDS using this A is:

$$\text{MSD}_{\text{diff,opt}}^{\text{network}} \approx \text{MSD}_{\text{diff,opt,k}} \approx \frac{\mu M}{4} \cdot \frac{1}{\sum_{l=1}^{K} \sigma_{v,l}^{-2}}\,, \tag{9.87}$$

To evaluate this we use the Chebyshev's sum inequality which states that if

$$a_1 \leq a_2 \leq .... \leq a_K$$

and

$$b_1 \geq b_2 \geq .... \geq b_K \,,$$

then

$$\frac{1}{K}\sum_{l=i}^{K} a_i b_i \leq \left(\frac{1}{K}\sum_{i=1}^{K} a_i\right)\left(\frac{1}{K}\sum_{i=1}^{K} b_i\right).$$

We can arrange the variances, $\{\sigma^2_{v,k}\}$, in order from smallest to largest as the $\{a_k\}$ in Chebyshev's sum inequality, this in turn would give the inverse variances, $\{\sigma^{-2}_{v,k}\}$, according to the $\{b_k\}$ and hence given this labeling the prerequisites for the inequality are met. For this choice of $\{a_k\}$ and $\{b_k\}$ the inequality looks instead as

$$\frac{1}{K}\sum_{l=1}^{K} \sigma^2_{v,l}\sigma^{-2}_{v,l} \leq \left(\frac{1}{K}\sum_{l=1}^{K} \sigma^2_{v,l}\right)\left(\frac{1}{K}\sum_{l=1}^{K} \sigma^{-2}_{v,l}\right).$$

But $\sigma^2_{v,l}\sigma^{-2}_{v,l} = 1$ making the left side equal to one and hence:

$$\left(\frac{1}{K}\sum_{l=1}^{K} \sigma^2_{v,l}\right)\left(\frac{1}{K}\sum_{l=1}^{K} \sigma^{-2}_{v,l}\right) \geq 1. \tag{9.88}$$

Keeping this in mind we now look at

$$\begin{aligned}\frac{1}{\sum_{l=1}^{K}\sigma^{-2}_{v,l}} &= \frac{\sum_{l=1}^{K}\sigma^2_{v,l}}{\left(\sum_{l=1}^{K}\sigma^2_{v,l}\right)\left(\sum_{l=1}^{K}\sigma^{-2}_{v,l}\right)} \\ &= \frac{1}{K^2}\frac{\sum_{l=1}^{K}\sigma^2_{v,l}}{\left(\frac{1}{K}\sum_{l=1}^{K}\sigma^2_{v,l}\right)\left(\frac{1}{K}\sum_{l=1}^{K}\sigma^{-2}_{v,l}\right)} \overset{(9.88)}{\leq} \frac{1}{K^2}\sum_{l=1}^{K}\sigma^2_{v,l}\,. \end{aligned} \tag{9.89}$$

So we have that

$$\frac{1}{\sum_{l=1}^{K}\sigma^{-2}_{v,l}} \leq \frac{1}{K}\cdot\left(\frac{1}{K}\sum_{l=1}^{K}\sigma^2_{v,l}\right),$$

which means that $\mathrm{MSD}^{\mathrm{network}}_{\mathrm{diff,opt}} \approx \mathrm{MSD}_{\mathrm{diff,opt,k}} \leq \mathrm{MSD}_{\mathrm{diff,sym,k}} \approx \mathrm{MSD}_{\mathrm{cent}}$. This states that when using optimal combination rules in diffusion methods we end up with a MSD that is as small if not smaller than than for the centralized method. This might seem strange, but notice that in the diffusion case we give nodes with better signal to noise ratio greater impact which of course gives a better result. Remember also that we can achieve estimates allowing us to use the Hastings rule without sending any data about the variances.

In General

So far we have only considered the limited case of ARMA-time series under the assumptions mentioned in the beginning of Section 9.2.5. The question is however these conclusions are valid for more general situations. In most cases we can apply the relative conclusions to the general ARMA-case, i.e., how the different MSD and convergence rates relate to each other. The derived numbers will not be true in general of course though.

Regarding the case of kernel methods, we should still expect similar results owing to the intuitive arguments presented trough this chapter. It is for example almost always true that the ATC algorithm performs better or at least not worse that the CTA. For the reader who is interested in learning more about convergence properties we refer to other literature. In (Haykin and Liu, 2009, Chapter 22) they do not assume uniform statistical distribution R_u. In (Sayed, 2012) they perform much of the analysis we have done here but rigorously and also considers more general cases such as using the algorithms in (9.71) and (9.72).

9.3 Conclusions

This chapter has discussed how we can achieve learning in WSN:s and has shown how we can use both kernel methods and ARMA-time series to this end. Different network topologies have been discussed and in the general topology we have introduced the cyclic algorithm, the diffusion methods and the alternating projection algorithm. All these algorithms all have their advantages and drawbacks. Some give very good results such as the general diffusion methods and others offer low communication needs, making them more applicable like the alternating projection algorithm and the local ATC and CTA algorithms. Ultimately what algorithm is suitable is very dependent on the situation but here we have introduced a wide variety of possible choices. Techniques described in this chapter have been used to learning regarding temperature fields, target localization, intruder detection, understanding biological networks, climate modeling and much more. See for example (Richard et al., 2010), (Sayed et al., 2013) and (Shanmuganthan et al., 2008). These are only some examples and the possibilities are nearly infinite making this area very important and interesting.

9.4 Consulted Material

This chapter has used several sources for the contained material. Some are already cited in the text, others are more of the general nature. In this section the materials that have produced this chapter are considered.

In Section 9.1.1, on supervised learning, most of the material is taken

from (Swami et al., 2007, Chapter 8), the examples are either inspired from other sources, Example 9.1.4 from (Predd et al., 2006a), or are of own construction.

Section 9.1.2, on ARMA-time series, includes material from (Haykin and Liu, 2009, Chapter 22), (Sayed et al., 2013) and (Sayed, 2012).

Section 9.1.3, on optimization, is inspired by more or less all the cited material, but primarily by (Haykin and Liu, 2009, Chapter 22).

Section 9.2.1, covering the star network, is primarily based on (Swami et al., 2007, Chapter 8), the examples however are inspired by (Predd et al., 2006a).

Section 9.2.2, on learning in general network topology, is a generalization of specific methods found in (Haykin and Liu, 2009, Chapter 22), (Sayed et al., 2013), (Richard et al., 2010), (Sayed, 2012) and (Swami et al., 2007, Chapter 8).

Section 9.2.3, learning in WSN using kernel methods, uses material from (Swami et al., 2007, Chapter 8), (Richard et al., 2010) and the material in Section 9.2.2 and thereby indirectly the there considered material.

Section 9.2.4, learning in WSN using ARMA-time series, is based on material from (Haykin and Liu, 2009, Chapter 22), (Sayed et al., 2013), (Sayed, 2012) and the material in Section 9.2.2.

Section 9.2.5, on convergence speed and precision, is based on material mainly from (Sayed et al., 2013), but also some from (Sayed et al., 2013), (Zhao and Sayed, 2012) and (Haykin and Liu, 2009, Chapter 22).

Problems

Problem 9.1 Motivate CTA

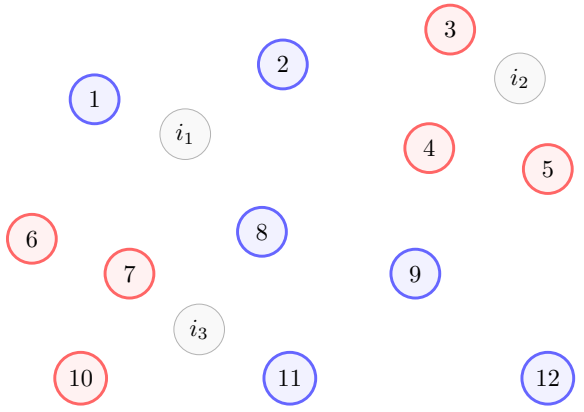
As seen in the part called *Motivation for Diffusion Methods* in section 9.2.2 we can formally motivate the usage of the ATC diffusion method. By following this derivation do your own derivation but instead for the CTA method.

Problem 9.2 ODE to ARMA-time series

As described in the part called *Applied to Differential Equations* in section 9.1.2 we can couple ODE:s and ARMA-time series. The following ODE describes the height position z, around the equilibrium position of $z = 0$, of a weight hanging in an fictional elastic spring subject to a small perturbation force $F(t)$:

$$m\frac{D^2}{Dt^2}z(t) + c\frac{D}{Dt}z(t) + kz(t) = F(t).$$

By using *Euler Backward*, found in (9.17) and (9.18), method convert this ODE into a ARMA-time series for predicting z. Show how ω^o looks in this model and assign the different terms in it to either the AR or MA part of the process. You can assume that the sampling time T is short enough for the approximation to be valid.



Sensor	Position (x_1, x_2)
1	$(-2.5, 2)$
2	$(0.2, 2.5)$
3	$(2.6, 3)$
4	$(2.3, 1.3)$
5	$(4, 1)$
6	$(-3.4, 0)$
7	$(0.2, -0.5)$
8	$(-0.1, 0.1)$
9	$(2.1, -0.5)$
10	$(-2.7, -2)$
11	$(0.3, -2)$
12	$(4, -2)$
i_1	$(-1.2, 1.5)$
i_2	$(3.6, 2.3)$
i_3	$(-1, -1.3)$

Figure 9.16 Illustration and table for **Problem 9.3**.

Problem 9.3 Using kernel methods

In this problem we consider the case where we get a set of training data from a WSN and using kernel methods would like to classify some points on a map. In Figure 9.16 the network is depicted. The nodes that are colored blue correspond to a positive measurement, i.e. 1, and the red ones to a negative measurement, i.e. 0. We would like to know how to classify the three nodes named i_1, i_2 and i_3. To do this use the kernel method in (9.4). Do this with the naive kernel (for $r_m = 4, 2, 1$) and the Gaussian kernel functions in Table 9.1. Try to see if you can predict what the three points will be classified as before doing the calculations.

Problem 9.4 Combination rules

Consider the network depicted in Figure 9.11. In this network suppose that we are going to employ some diffusion strategy. This requires a set of combination weights $\{a_{kl}\}$. Now consider some different combination rules and compute what weights they would generate. Calculate the weights corresponding to node 0, 2 and 14, i.e. $a_{0,l}$, $a_{2,l}$ and $a_{14,l}$ for all l, for the combination rules stated below.

a) The maximum degree Laplacian rule in (9.46);

b) The Metropolis rule in (9.48);

c) The relative degree rule in (9.49).

Chapter 10

Positioning and Localization

WSNs have the capability to monitor phenomena in the physical world, establish spatial relationships between the nodes, and detect objects and events; hence offering many convenient services. However, without being able to know the position of a sensor node, or of an object or person that has to be tracked in its spatial position, the usage of WSN is limited to a great extent. For example WSNs deployed in a forest to raise alarms whenever wildfires occur, may play a crucial role if they are able to report the spatial relationship between them and the monitored event. Further, various other tasks rely on accurate location information, such as location-aware services, surveillance systems, motion tracking, and many others.

Localization is the process of determining the physical coordinates of a sensor node or the spatial relationships among objects. There are many techniques to achieve this goal, a sensor node can collect information from the surroundings to estimate its position. For example, from ranging measurements such as RSS, the distances or ranges between a number of transmitters and a received node could be measured and used to find the position of the receiver node. Every localization procedure is ultimately based on detection and estimation theory, and on distributed estimation.

This chapter begins with an introduction of positioning and localization. Then various ranging techniques are discussed. There are two main types of localization i.e. range-based and range-free localization. These methods are presented in the following section.

10.1 Introduction

Localization can be generally divided into two classes, range-based techniques and range-free techniques. Range-based techniques are based on distance measurement using ranging techniques (for example, the received signal strength, angle of arrival, and time of arrival). They require presence of at least three special nodes, called "anchor nodes" whose positions are clearly

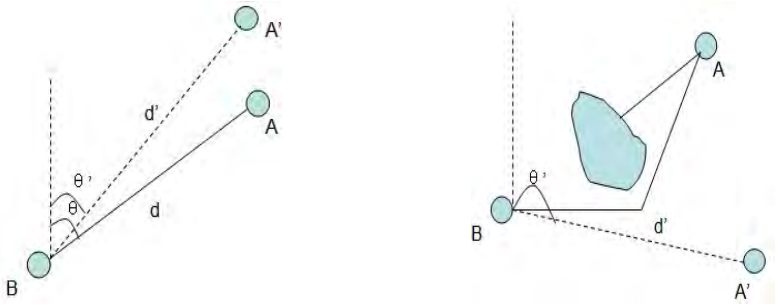


Figure 10.1 Measurement errors.

known by all the nodes in the network. The position of unknown nodes can be estimated using the distance measurements between them and anchor nodes. In contrast, range-free techniques do not require distance measurements, they use connectivity information to determine the positions. Triangulation, Trilateration, Iterative and Collaborative Multilateration are some examples of range-based localization. Ad-Hoc Positioning System, Approximate Point in Triangulation and Localization Based on Multidimensional Scaling fall in the category of range-free localization.

10.2 Challenges

Localization faces many challenges such as physical layer measurement errors, computational constraints, lack of GPS data, low-end sensor nodes, and varying configurations. In addition, different applications have different requirements. A WSN localization system will come across various challenges to fulfill all kinds of requirements. The main challenges in WSNs are discussed below.

10.2.1 Physical Layer Measurements

Sensor nodes can use ranging techniques to estimate their positions. The ranging measurements can be time, angle or received signal strength. These measurements can affect the accuracy of the estimated positions greatly. For line-of-sight communication, small error in the measurement leads to large deviation in the estimated position. For example, as shown in the Figure 10.1 node A transmits a signal to node B in order to estimate its position using received signal strength. If the estimated angle $\hat{\theta}$ varies only a little from the real θ and the radius is also large, the estimated position differs a lot from the actual one. In addition, if there is an obstacle between A and B, multipath propagation can induce even larger error in the estimates.

10.2.2 Computational Constraints

A node collect information in terms of distance, time, orientation or connectivity from neighbors to estimate its positions. In order to find the exact location, a node should combine as much information as possible to do the estimation, this requires implementation of algorithms that may require a large computational complexity. Due to the limitation of a node's memory length and processing ability, some computations may be done at a particular processing node. This will add information overhead and increase transmission delay.

10.2.3 Lack of GPS

GPS(Global Positioning System) is widely used to determine position in many systems such as navigation and phone. It can provide precise positioning outdoor, but indoor may give high unreliability. In addition, it is not feasible to use GPS in all of the nodes in a WSN. The reason being the high cost and large power consumption requirement for a node. However, some of the anchor sensor nodes can be equipped with GPS. These anchor nodes may then use the GPS as a mean to initialize positioning algorithms with the GPS information.

10.2.4 Low-End Sensor Node

Wireless sensor nodes may be equipped with low-end components to provide low-cost operation. These imperfect components pose several challenges for localization in WSNs. In addition to errors in range measurement, some hardware errors can also be introduced into the measurement processes. These errors are random, we can do little to avoid them. Low-end component can cause node failure frequently. All of these challenge the accuracy of position estimations.

10.3 Ranging Techniques

Localization techniques often rely on the measurement of the distance between the nodes to calculate positions. These distances can be measured by considering certain characteristics of the signal such as signal strength, angle of arrival, propagation time etc. Since such methods are used to measure the distance between nodes, they are called ranging techniques. A few of these techniques are discussed below.

10.3.1 Time of Arrival

The key principle of Time of Arrival (ToA) method is to determine the distance between the sensor nodes using measured signal propagation time

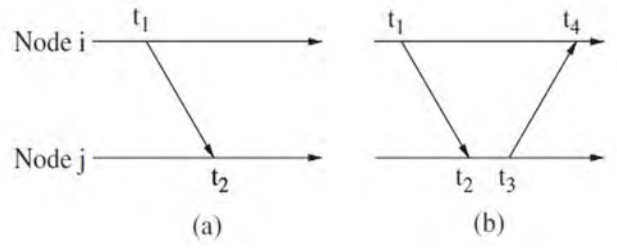


Figure 10.2 One-way and two-way ToA ranging measurement scheme.

and known propagation velocity. ToA has two types, one-way ToA and two-way ToA. One-way ToA measures the one-way signal propagation time and requires the sender and the receiver to be synchronized with each other. The difference between the sending time and receiving times is calculated as shown in Figure 10.2(a). The distance between the nodes i and j can be determined as

$$d_{ij} = (t_2 - t_1) \times v \tag{10.1}$$

where t_1 and t_2 are the sending and receive times of the signal (measured at the sender and receiver, respectively). Here, the receiver calculates the distance and uses it to determine its location.

In two-way ToA, in addition to the first signal, the receiver then transmits a response signal back to transmitter. This is shown in Figure 10.2(b). So we have four time points and the transmitter uses them, together with signal velocity, to measure the distance.

$$d_{ij} = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} \times v\,, \tag{10.2}$$

where t_3 and t_4 are the sending and receive times of the response signal. Hence the transmitter is calculating the receiver's location. Note that a third message is necessary to inform the receiver about its location. Moreover this two-way technique does not require synchronization of the sender and the receiver, hence making it a preferred approach.

10.3.2 Time Difference of Arrival

The time difference of arrival (TDoA) approach uses two signals that travel with different velocities.

As shown in Figure 10.3, the transmitting node sends a signal with speed v_1 at time t_1, the receiving node receives this signal at time t_2. After a time delay $t_{\text{delay}} = t_3 - t_1$, the transmitter sends another signal with velocity v_2, the receiver gets this signal at time t_4. Then we can measure the distance

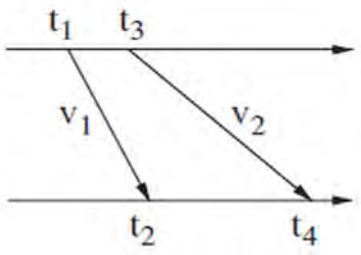


Figure 10.3 TDoA measurement scheme.

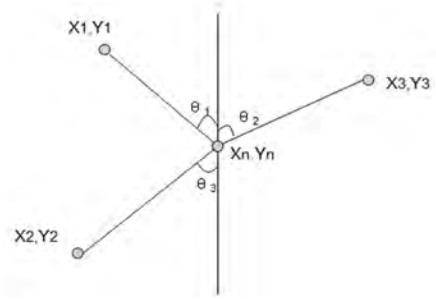


Figure 10.4 Angle of Arrival ranging measurement scheme.

between transmitter and receiver using these measurements.

$$d_{ij} = (t_4 - t_2 - t_{\text{delay}}) \times (v_1 - v_2) \ . \tag{10.3}$$

One advantage of TDoA is that it does not require time synchronization between transmitter and receiver. The estimation of TDoA may have a better accuracy compared to ToA, but it can require additional hardware, for example, a microphone and speaker if one intend to use acoustic signal as one of the two signals.

10.3.3 Angle of Arrival

A node can estimate its position by measuring angles of arriving signals using an array of antennas or microphones. Measurements are made from at least three anchor nodes as shown in Figure 10.4.

This mechanism can provide precise localization, but it depends on the accuracy of directional antennas, which may not be easy have inside a sensor node. The addition of measurement hardware can raise the cost and size of a sensor node. Also, this method can easily get corrupted in NLoS environment due to multi-path fading and scattering, which prevent the accurate measurement of the angles.

10.3.4 Received Signal Strength

The method of measuring received signal strength relies on that a transmitted signal power decays with distance. Received Signal Strength Indicator (RSSI) is a common feature in wireless devices that can measure the power of the incoming radio signal. The RSSI is a mapping of the power into quantized levels. The mapping between the RSSI value and signal power varies from vendor to vendor. The distance can be calculated according to the received signal power. In the simplest case of no attenuations due to slow and fast fading, the Friis transmission equation for free space gives

$$\frac{P_r}{P_t} = G_t G_r \frac{\lambda^2}{(4\pi)^2 d^2}\,, \tag{10.4}$$

where G_t is the antenna gain of the transmitting antenna and G_r is the antenna gain of the receiving antenna. In reality, the signal power is affected by multi-path propagation, noise and so on, so this equation gives a rough ideal approximation.

10.4 Range-Based Localization

10.4.1 Triangulation

This technique relies on the geometric relationship between unknown nodes and anchor nodes. The position of the unknown node is estimated by measuring the angle of arrival of signals from anchor nodes. Then some statistical method (for example, maximum likelihood algorithm) is used to minimize the estimation error. This is illustrated in Figure 10.5 where we have three anchor nodes with known locations $\mathbf{x}=[x_i, y_i]^T$ where i=1,2,3 and one unknown node at location $\mathbf{x}_r=[x_r, y_r]^T$. The actual angles between the unknown node and anchors are

$$\theta(X_r) = [\theta_1(X_r), ..., \theta_N(X_r)]^T\,,$$

where N=3 and

$$\theta_i\,(X_r) = \arctan\frac{x_r - x_i}{y_r - y_i}\,.$$

But due to some noise or errors in the measurement process, the measured angles do not perfectly reflect the actual angles and are represented as $Y = [\tilde{\theta}_1, ..., \tilde{\theta}_N]^T$, so that we have

$$Y = \theta(X_r) + \underline{n}\,, \tag{10.5}$$

where $\underline{n} = [n_1, n_2, n_3]^T$ is Gaussian noise with zero mean and covariance given by

$$R = \mathbb{E}\left\{n \cdot n^T\right\} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}\,. \tag{10.6}$$

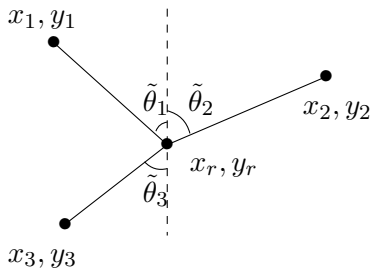


Figure 10.5 Measurements for the triangulation scheme.

If we use ML criterion to estimate a sensor's location, the location estimation is achieved by minimizing the following error covariance

$$C(X_r) = \left[\theta(\hat{X}_r) - Y\right]^T R^{-1} \left[\theta(\hat{X}_r) - Y\right] \tag{10.7}$$

$$= \sum_{i=1}^{3} \frac{\left(\theta_i(\hat{X}_r) - \tilde{\theta}_i\right)^2}{\sigma_i^2}, \tag{10.8}$$

and hence

$$\hat{X}_r = \text{argmin} \quad C(X_r) \tag{10.9}$$

To find the position estimation ($\hat{X}_r = [\hat{x}_r, \hat{y}_r]^T$), we have to minimize $C(X_r)$. This can be achieved by taking a derivative of $C(X_r)$ and equating it to zero. Since this is the minimization of a non-linear least-square, we apply the Newton-Gauss method:

$$\begin{aligned}\hat{X}_{r,i+1} =& \hat{X}_{r,i} \\ &+ (\theta_X(\hat{X}_{r,i})^T S^{-1} \theta_X(\hat{X}_{r,i}))^{-1} \theta_X(\hat{X}_{r,i})^T S^{-1} [Y - \theta_X(\hat{X}_{r,i})],\end{aligned} \tag{10.10}$$

where $\theta_X(\hat{X}_{r,i})$ is the matrix of the partial derivatives of θ with respect to its arguments and evaluated at $\hat{X}_{r,i}$. As i tends to ∞, $\hat{X}_{r,i}$ tends to the minimum of $C(X_r)$. Equation (10.10) requires an initial estimate that is close to the true minimum of the cost function.

10.4.2 Trilateration

Trilateration is the process of calculating a node's position based on the measured distance between this node and other anchor nodes whose positions are known. Obviously, for a given distance, this node must be positioned on the circumference of a circle centered at an anchor node with a radios given by the distance between these two nodes. In Figure 10.6 anchor nodes are

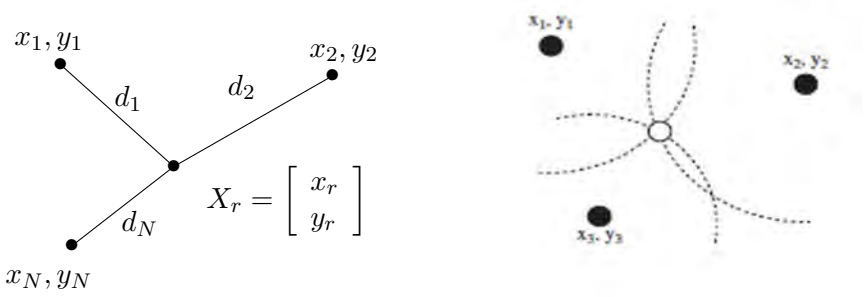


Figure 10.6 Trilateration scheme.

located at $\mathbf{x}_i = (x_i, y_i)(i = 1, ..., N)$ and the unknown node is at location $X_r = [x_r, y_r]^T$. The distance measurements are assumed to be corrupted by noise and given by

$$\widetilde{d}_i = d_i + n_i\,, \qquad i = 1, ..., N\,. \tag{10.11}$$

From simple trigonometry,

$$\begin{cases} (x_1 - x_r)^2 + (y_1 - y_r)^2 = \widetilde{d}_1^2 \\ \vdots \\ (x_N - x_r)^2 + (y_N - y_r)^2 = \widetilde{d}_N^2\,. \end{cases} \tag{10.12}$$

After subtracting these N equations we arrive at the following system of equations

$$A \cdot X_r = Y\,, \tag{10.13}$$

where

$$A \in \mathbb{R}^{(N-1)\times 2}, \quad X_r \in \mathbb{R}^2, \quad Y \in \mathbb{R}^{(N-1)\times 1}\,,$$

$$A = 2 \cdot \begin{bmatrix} (x_N - x_1) & (y_N - y_1) \\ \vdots & \vdots \\ (x_N - x_{N-1}) & (y_N - y_{N-1}) \end{bmatrix}, \tag{10.14}$$

$$Y = \begin{bmatrix} \widetilde{d}_1^2 - \widetilde{d}_N^2 - x_1^2 - y_1^2 + x_N^2 + y_N^2 \\ \vdots \\ \widetilde{d}_{N-1}^2 - \widetilde{d}_N^2 - x_{N-1}^2 - y_{N-1}^2 + x_N^2 + y_N^2 \end{bmatrix}. \tag{10.15}$$

We can the get the estimation of position as follows:

$$\hat{X} = LY\,, \tag{10.16}$$

where $L = \left(A^T A\right)^{-1} A^T$.

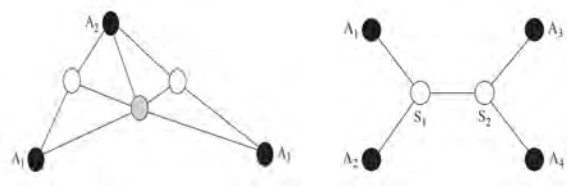


Figure 10.7 (a) Iterative multilateration (b) Collaborative multilateration

10.4.3 Iterative and Collaborative Multilateration

Iterative and collaborative multilateration is an extension of lateration technique that requires at least three anchor nodes to determine position of the fourth unknown node. With these extended multilateration techniques it is possible to estimate position even without the presence of three neighbouring anchor nodes. If a node has determined its position then it becomes an anchor node and broadcasts its anchor message to other nodes. This process continues until all the nodes in a network have been localized. It is called *iterative multilateration* and shown in Figure 10.7(a); three black nodes aid in localization of the gray node in first iteration and in second iteration, white nodes estimate their respective locations with the help of the gray node and two black anchor nodes. The drawback of this technique is that estimation error accumulates with each iteration.

It is possible that all the nodes will have less than three neighbouring anchor nodes, in this case *collaborative multilateration* is adopted to estimate position. In the Figure 10.7(b), there are 4 anchors and 2 unknown position nodes. The goal of collaborative multilateration is to construct a graph of participating nodes, that is, nodes that are anchors or have at least three participating neighbours. As a result a set of over-constrained quadratic equations is obtained that relate the distance among nodes and its neighbours. These equations are solved to estimate the positions in the network.

10.5 Range-Free Localization

Range-free localization techniques does not rely distance estimation using ranging techniques, instead it uses connectivity information for estimating the position. So, these techniques do not require installation of additional hardware on the nodes that makes them cost-effective alternative to the range-based algorithms. One such technique is called Ad Hoc Positioning System (APS) which is discussed next.

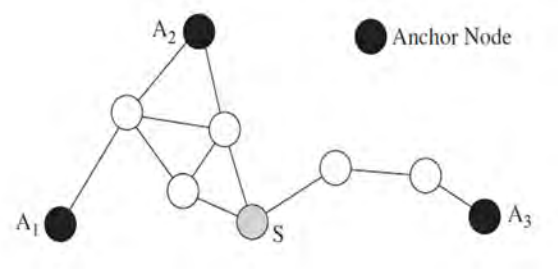


Figure 10.8 APS:DV-hop localization.

Ad Hoc Positioning System (APS)

Basic principle

APS is the most typical model of distributed localization based on connectivity between nodes in the network. In this algorithm anchor nodes are also needed to estimate unknown positions, in-fact, the accuracy of estimation increases by increasing the number of anchors. At least three anchor nodes are required for localization. Each anchor node propagates its position to other nodes using Distance Vector. All the nodes exchange their routing tables with one-hop neighbours. In the most basic scheme of APS, called DV-hop, each node stores a table $\{X_i, Y_i, h_i\}$ where $\{X_i, Y_i\}$ means the location of anchor node i and h_i is the distance in hops between this node and node i. When an anchor node knows the location of other anchor nodes and the distance in hops between them, it can determine the average size of one hop called the correction factor. The correction factor c_i of anchor node i is:

$$c_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_i} \quad (j \neq i) \ . \tag{10.17}$$

Correction factors are propagated in the network via controlled flooding which means that each node uses only one correction factor, usually the first from the closest neighbour. When a node knows the distance in hops between it and anchors nodes (at least 3) and the correction factor, it can implement trilateration to estimate its own location.

For example in the network shown in Figure 10.8, if $d(A_1, A_2) = 50$, $d(A_2, A_3) = 60$, $d(A_1, A_3) = 80$, the correction factor of A_1 will be $c_1 = (50 + 80)/(2 + 6) = 16.25$. The same $c_2 = (50 + 60)/(2 + 5) = 15.71$, $c_3 = (60 + 80)/(6 + 5) = 12.73$. Node S has A_2 as the closest anchor so it will use correction factor $c_2 = 15.71$. It can calculate the distances between itself and anchor nodes by multiplying c_2 with number of hops with each of the three anchor nodes (3x15.71 to A_1, 2x15.71 to A_2, 3x15.71 to A_3). Then trilateration can be used to find its location in the network.

Improvements

Localization through APS can have some errors, however, some improvements are suggested in order to minimize the estimation error.

1. When each node gets the one-hop distance, use the formula

$$\text{Hopsize}_{\text{ave}} = \frac{\sum \text{Hopsize}_i}{n},$$

where n is the number of anchor nodes, Hopsize_i denotes the one-hop distance calculated from anchor i.

2. Increase the number of anchor nodes, the more anchors ensures more reliable results.

3. We can analyze the deviation between each anchor's one-hop distance and the average one-hop distance, then modify the average one-hop distance we previously got. The following steps will show how it works. In the beginning, we define the average one-hop distance deviation of anchor i as

$$\text{err\_dis}_i = \sum_{i \neq j} \frac{|d_{\text{true}} - d_{\text{estimate}}|_{ij}}{\text{hops}_{ij} \times n_i} \tag{10.18}$$

Here, hops_{ij} means number of hops between anchor i and j, n_i means the number of anchors in anchor's data list,

$$d_{\text{true}} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

where

$$d_{\text{estimate}} = \text{Hopsize}_{\text{ave}} \times \text{hops}_{ij}.$$

First, we calculate the average one-hop distance, then we use Equation (10.18) to compute the average one-hop distance deviation and propagate it to the network in the form of $\{\text{id}, \text{err\_dis}_i\}$. Secondly, every node receiving this data package stores the information in its data list and transmit it to its neighbors, the package with the same id number will be dropped. Thirdly, when every node acquires the average one-hop deviation of each anchor, we can do the calculation $c\_\text{err\_dis} = \sum \text{err\_dis}_i / n$, where n is the number of anchors. Finally, we can recalculate the average one-hop distance:

$$\text{New\_Hopsize}_{\text{ave}} = \text{Hopsize}_{\text{ave}} + k \times c\_\text{err\_dis}, \tag{10.19}$$

where k is from -1 to 1.

4. Any network has its region, as the estimated distance between node and anchor has some variations from the true one, the estimated position may be located outside the network region which increases the average location errors. We can re-modify the locations that are out of the regions. If $X \leqslant X_{\min}$, we make $X = X_{\min}$. If $X \geqslant X_{\max}$, we make $X = X_{\max}$, the same with Y axis.

Problems

Problem 10.1 Timing Offset and GPS (Ex.9.1 in (Pottie and Kaiser, 2005))

GPS uses a constellation of 24 satellites and their ground stations as reference points to calculate positions accurate to a matter of meters. Suppose we find our distance measurements from three satellites to be 18 000, 19 000, and 20 000 km respectively. Collectively this places the location at either of the two points where the 20 000 km sphere cuts through the circle that is the intersection of the 18 000 and 19 000 km spheres. Thus by ranging from three satellites we can narrow our position to just two points in space. To decide which one is our true location we could make a fourth measurement. However, usually one of the two points is a non-possible answer (either too far from Earth or moving at an impossible velocity) and can be rejected without a measurement. Now apply the above principle of location in a two-dimensional space. Assume that points A, B, and C are reference points with known locations, respectively at (x_1, y_1), (x_2, y_2), and (x_3, y_3), and that the unknown position is 3.0 meters from point A, 4.0 meters from point B, and 5.0 meters from point C.

(a) Suppose that accurate measurements are available. Then the three measurements can be used to uniquely determine the position. Let $(x_1, y_1) = (0, 3.0)$, $(x_2, y_2) = (4.0, 0)$, $(x_3, y_3) = (4.0, 3.0)$. Find the position.

(b) Now assume that all measurements include a single timing offset that corresponds to an error of 0.5 m. In other words, the position is observed to be 3.5 m from point A, 4.5 m from point B, and 5.5 m from point C. Develop a generic procedure to find the true position.

Problem 10.2 Linearizing GPS Equations (Ex.9.2 in (Pottie and Kaiser, 2005))

In order to find position using the GPS system, we need to know the location of at least three satellites and the distance to each of those satellites. Assume that the three satellites are located respectively at (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3), and that the distance between us and the three satellites are respectively $d1, d2, d3$. The following nonlinear system of equations needs to be solved,

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 &= d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 &= d_2^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 &= d_3^2 \,. \end{aligned} \tag{10.20}$$

Obviously linearization is desirable in this case. Assume that the reference point is $(0, 0, 0)$. Prove that the resulting system after linearizing (10.20) is

$$2 \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & y_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1^2 + y_1^2 + z_1^2 - d_1^2 \\ x_2^2 + y_2^2 + z_2^2 - d_2^2 \\ x_3^2 + y_3^2 + z_3^2 - d_3^2 \end{bmatrix}$$

.

Problem 10.3 Averaging to reduce error in TOA (Ex.9.3 in (Pottie and Kaiser, 2005))

TOA is based upon the measurement of the arrival time of a signal transmitted from the to-be-located object to several reference nodes. For radio signals, the distance is ct, where c is the velocity of light and t is time of travel from the object to the reference node. This measurement thus indicates the object is on a circle of radius ct, centered at the reference node. There is always a need for at least three reference nodes to determine the location of the object correctly. The disadvantage of this technique is that processing delays and non-line-of-sight propagation can cause error, resulting in mistakes in the TOA estimation. Assume that t is a Gaussian distributed RV with mean at the real time of arrival $\bar{t}$ and a variance δ_t.

(a) Find the mean and variance of the resulting range of the object.

(b) Now assume that independent multiple measurements of range are available. That is, $t(n)$, $n = 1, 2, 3, ..,$ is the measured time of arrival from the reference node to the to-be-located object, at time instant n. Show that multiple measurements help to reduce the error in the resulting range of the object.

Problem 10.4 Weighted centroid computation (Ex.9.9 in (Pottie and Kaiser, 2005))

Three beacons are located at $a = (1, 1)$, $b = (1, -1)$, and $c = (-1, 1)$. The received powers from nodes a, b, and c are 1.2, 1.5, and 1.7 respectively. Calculate the unknown position of the receiver through a weighted centroid computation.

Problem 10.5 Collaborative multilateration

Consider Figure 10.9, suppose node U can estimate ranges only for nodes A, C, and V, and node V can estimate ranges only for nodes B, D, and U, where the unknown locations are U and V. One can begin with an initial guess at the position of U from either the centroids of the known positions in immediate range, or via the topology. Then multilateration is performed using the locations of all neighbors (estimated or known) to refine the positions, in a sequence that proceeds until locations stabilize. Compute the first estimate of the positions of $U(u_0)$ and $V(n_0)$ as the centroids of the nodes they can hear that have known position. Then iteratively calculate by multilateration the positions in the order u_1, n_1 assuming perfect range measurements.

Problem 10.6 Linearization of angle of arrival (AOA) location determination (Ex.9.11 in (Pottie and Kaiser, 2005))

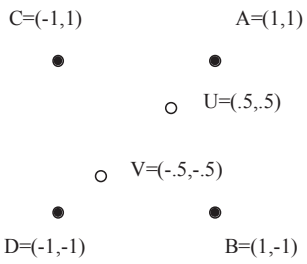


Figure 10.9 Four node multilateration.

The intersection of the angles from two or more sites may be used to provide an unknown location in the plane. For this triangulation problem, denote the position of the two known nodes as $r_i = [\ x_i \quad y_i\ ]^T, i = 1, 2$, and the unknown node's position as $r = [\ x \quad y\ ]^T$. The bearing angles can be expressed as

$$\theta_i = f_i(r, r_i) + n_i, \quad i = 1, 2, \tag{10.21}$$

where n_i is the angle measurement error, and the function $f_i()$ is defined as

$$f_i(r, r_i) = \arctan\left(\frac{x - x_i}{y - y_i}\right), i = 1, 2. \tag{10.22}$$

After collecting angle measurements from known nodes, the unknown node's position can be found by solving the nonlinear system of equations

$$\begin{aligned} \theta_1 &= \arctan\left(\tfrac{x-x_1}{y-y_1}\right) + n_1 \\ \theta_2 &= \arctan\left(\tfrac{x-x_2}{y-y_2}\right) + n_2\,. \end{aligned} \tag{10.23}$$

This triangulation problem can alternatively be solved by linearizing the $f_i()$ function by expanding it in a Taylor series around a reference point, denoted by r_0. Once the equation system is linearized, the ML estimator is used to provide the following unknown node position estimate

$$\begin{aligned} \hat{r} &= r_0 + \left(G^T N^{-1} G\right)^{-1} G^T N^{-1} \begin{bmatrix} \theta_1 - f_1(r_0) \\ \theta_2 - f_2(r_0) \end{bmatrix} \\ &= r_0 + G^{-1} \begin{bmatrix} \theta_1 - f_1(r_0) \\ \theta_2 - f_2(r_0) \end{bmatrix}. \end{aligned} \tag{10.24}$$

Matrix $N = E[nn']$ is the measurement error covariance matrix, and matrix G is the matrix of the resulting equation system after linearizing (10.21). Matrix G is equal to

$$G = \begin{bmatrix} (y_0 - y_1)/d_{01}^2 & -(x_0 - x_1)/d_{01}^2 \\ (y_0 - y_2)/d_{02}^2 & -(x_0 - x_2)/d_{02}^2 \end{bmatrix},$$

where angle $\theta_{0i} = f_i(r_0), i = 1, 2$, and d_{0i} is the distance between the i th node and r_0. Given $r_0 = [\ 0 \quad 0\ ]^T$, $r_1 = [\ -3 \quad 4\ ]^T, r_2 = [\ 4 \quad 3\ ]^T$, $\theta_1 = 45°, \theta_2 = 135°$, and

$$N = \begin{bmatrix} 1 & 0 \\ 0 & 0.9 \end{bmatrix}.$$

Use equation (10.24) to find the unknown node's position. Comment on the accuracy of the results.

Chapter 11

Time Synchronization

In the previous chapter the topics of localization and positioning of sensor nodes within a WSN were covered. Various techniques are applied to retrieve the location information of the nodes which is associated with their gathered information to provide an accurate view of the observed sensor field. To determine where and when the events occur and how they evolve in space and time, the sensors must know their own position and the time. Many techniques for determining location in turn depend on having precise time references. Moreover, since an event is usually determined in its entity by the collaborative information from multiple nodes, timing information among these nodes needs to be consistent. Thus the two topics of localization and synchronization are closely connected.

Time synchronization also ensures compatibility in terms of protocol development. As an example, time division multiple access (TDMA) protocols require neighbor nodes to be synchronized so that they can follow a common time frame for medium access. As explained in Chapter 4, sensor nodes following a TDMA protocol need to agree on boundaries of time slots; otherwise their transmissions would overlap and collide. Furthermore, with respect to energy, many WSNs rely on sleep/wake protocols that allow a network to selectively switch off sensor nodes or let them enter low-power sleep modes. Therefore, temporal coordination among sensors is essential for nodes to know when they can enter a sleep mode and when to rewake to ensure that neighboring nodes overlap in their wake periods to enable communication among them.

In this chapter, the basic topics about time synchronization in WSNs are covered. In general, a WSN nodes is equipped with its own local clock for internal operations. Since each node operates independently on its own clock and considering possible random phase shifts and drift rates of oscillators, the local time reading of nodes differ. Moreover, the multi-hop and packet-based information delivery in WSNs results in variations in the information delivery time. The delay between a node and the sink is proportional to the

distance between them. Consequently, the received time of the packets at the sink and the order in which they are received do not correctly represent the sensing time of the events. Time synchronization is therefore essential for a seamless network operation.

The chapter is organized as follows: Section 9.1 provides the basic definitions regarding the sensors' clocks and the challenges that occur in the time synchronization process. Next, in Section 9.2, the basic aspects regarding the time synchronization process are covered. In particular, the way nodes exchange their synchronization messages is examined in detail. In Section 9.3, the most representative time synchronization protocols are presented while the last section of the chapter describes a fully distributed protocol, the Gradient Time Synchronization Protocol.

11.1 Node Clocks and Synchronization Problem

In this section we give some introductory concepts that are used for node synchronization protocols.

A typical node possesses an **oscillator** of a specified frequency and a **counter register**, which is incremented in hardware after a certain number of oscillator pulses. The node's software has only access to the value of this register and the time between two increments (ticks) determines the achievable **time resolution**. The value of the **hardware clock** of node i at real time t can be represented as $H_i(t)$. Further, the **software clock** $C_i(t)$ of node I at some real time t is given by

$$C_i(t) = \rho_i(t) \cdot H_i(t) + \phi_i(t)\,, \tag{11.1}$$

where $\phi_i(t)$ is called **phase shift** and $\rho_i(t)$ is called **drift rate**. In a perfect clock $\rho_i(t) = 1$ and $\phi_i(t) = 0$. Clock adjustment is performed by properly adjusting the parameters ϕ_i and ρ_i since it is often neither possible nor desirable to influence the oscillator or the counter register. In WSNs, the low-cost crystals oscillators introduce both drift rate, i.e., $\rho_i(t) \neq 1$, and phase shift, i.e., $\phi_i(t) > 0$. Since these parameters can be different for each sensor node, the nodes result unsynchronized. Note that the phase shift and drift rate of a local clock shown in Equation 11.1 are also function of time.

The frequency at which a clock progresses is called clock rate, namely dC/dt. The maximum drift rate of a clock is expressed as ρ and for the clock rate the following condition holds:

$$1 - \rho \leq \frac{dC}{dt} \leq 1 + \rho\,, \tag{11.2}$$

Further, comparing the local software times of two nodes i and j, the **clock offset** indicates the difference between the times. Synchronization is

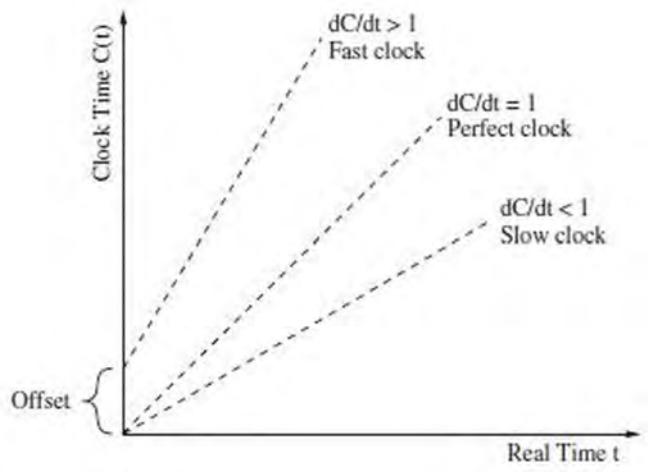


Figure 11.1 Relationship between software time $C(t)$ and real time t.

then defined as the procedure of adjusting properly the time of at least one of these node clocks such that their readings match.

Oscillators often have a priori a slight random deviation from their nominal frequency, called **drift** or **clock skew**. This can be due to impure crystals but oscillators also depend on several environmental conditions like pressure, temperature, and so on, which in a deployed WSNs might well differ from laboratory specifications. The clock drift is often expressed in **parts per million(ppm)** and gives the number of additional or missing oscillations a clock makes in the amount of time needed for one million oscillations at the nominal rate.

Perfect or ideal clocks are those with clock rate $dC/dt = 1$ at all times. However, as shown in Figure 11.1, the clock rate can take a value different from 1 due to environmental factors or hardware flaws that affect the actual clock rate resulting in a drift rate.

Figure 11.1 illustrates how the drift rate affects the clock reading with respect to real time, resulting in either a perfect, fast, or slow clock. This drift rate is responsible for inconsistencies in sensors' clock readings even after clocks have been synchronized, making it necessary to repeat the synchronization process periodically.

Assuming two different nodes with synchronized clocks, their clocks can drift from each other at a rate of at most $2\rho_{\max}$. Therefore, in order to find out when a resynchronization is needed, the following inequality can be used:

$$\frac{dC_i}{dt} - \frac{dC_j}{dt} \leq 2\rho_{\max}\,. \tag{11.3}$$

The delay difference between two node clocks is usually categorized by an initial time offset t_0, a time-dependent frequency offset $\Delta f(t)$, and jitter

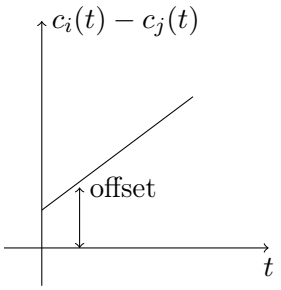


Figure 11.2 Relationship between software time $C(t)$ and real time t.

$\Delta\tau(t)$ due to noise, as follows:

$$c_i(t) - c_j(t) = t_0 + \Delta f(t) \cdot t + \Delta\tau(t). \tag{11.4}$$

The frequency offset is due to both differences in the hardware details of different clocks as well as environmental conditions such as temperature of operation, humidity etc. A clock is said to be stable if the frequency offset changes slowly. The relative phase between clocks can be obtained by differentiating with respect to delay. Figure 11.2 shows that the difference $C_i(t) - C_j(t)$ can become arbitrarily large as t increases. Therefore, a time synchronization protocol is needed. The objective of a time synchronization protocol is to establish a one-to-one correspondence between the times reported by the different clocks.

In general, we distinguish two types of synchronization: *external* and *internal (or distributed)*. External synchronization means that the clocks of all nodes are synchronized with an external source of time (or *reference clock*). The external reference clock is an accurate real-time standard such as Coordinated Universal Time (UTC).

Internal (or distributed) synchronization means that the clocks of all nodes are synchronized with each other, without the support of an external reference clock. The goal of internal synchronization is to obtain a consistent view of time across all nodes in the network, even though this time may be different from any external reference times. External synchronization ensures both synchronization with an external source and consistency among all clocks within the network. When nodes are synchronized to an external reference clock, the *accuracy* of a clock describes the maximum offset of a clock with respect to the reference clock. In particular, nodes $1, 2, \ldots, n$ are said to be accurate at time t within a bound δ if $|C_i(t) - t| < \delta$ holds for all nodes $i \in \{1, 2, \ldots, n\}$.

On the other hand, when nodes in a network are internally synchronized, the precision indicates the maximum offset between any two clocks in the network (Kopetz 1997). The nodes $1, 2, ..., n$ are said to agree on the time with a bound of δ if $|C_i(t) - C_j(t)| < \delta$ holds for all $i, j \in 1, 2, \ldots, n$.

Clearly, if two nodes are externally synchronized with an accuracy of δ, they are also internally synchronized with a precision of 2δ.

11.1.1 Challenges for Time Synchronization

Traditional time synchronization protocols proposed in wired networks cannot be directly applied to WSNs without considering the specific challenges that occur in low-cost low-power sensor nodes and the wireless medium. Similar to wired environments, time synchronization in WSNs needs to face challenges such varying clock drifts due to changes in temperature and humidity. However, the nature of a WSN imposes a set of additional challenges and constraints that time synchronization protocols for sensor networks have to consider. Some of these challenges are summarized below:

Environmental Effects

Environmental factors such as temperature, pressure, and humidity affect the clock readings. While typical wired network devices are operated in rather stable environments (e.g., A/C-controlled cluster rooms or offices), wireless sensors are frequently placed outdoors and in harsh environments where these fluctuations in ambient properties are common.

Energy Constraints

As mentioned in the previous chapters, energy consumption and the resulting constraints are of crucial importance in a WSN. Wireless sensor nodes are typically driven by finite power sources, that is, either disposable or rechargeable batteries. Battery replacement can add significantly to the cost of a WSN, particularly in large-scale networks and when the nodes are in difficult-to-service locations. Therefore, time synchronization protocols should not contribute significantly to the energy consumption of wireless nodes in order to ensure long battery life times. Since communication among sensor nodes is typically the basis for time synchronization, an energy-efficient synchronization protocol should aim to minimize the mutually transmitted messages that are necessary to obtain synchronized nodes.

Wireless Medium and Mobility

Synchronization requires nodes to communicate with each other to exchange clock information through which the local clocks can be synchronized. While this communication is usually trivial in wired networks such as the Internet, WSNs require wireless communication between nodes, which creates additional challenges for synchronization as a result of the error-prone communication and non-deterministic delays. Firstly, the wireless channel errors result in some of the synchronization messages being lost. Thus, some

nodes in the network may be unsynchronized. More importantly, the synchronization messages sent by the unsynchronized nodes force other nodes to adapt to their local clocks. Therefore, robust synchronization methods are required. Secondly, the broadcast nature of the wireless channel necessitates MAC protocols being utilized for efficient channel access. These MAC protocols introduce a non-deterministic access delay, which is the time between the synchronization protocol issuing a synchronization packet to be sent and the time this packet is actually transmitted. As we will see next, the channel access operation introduces an important randomness in time synchronization and needs to be accounted for in calculations. Finally, the wireless channel introduces an asymmetric delay between two nodes for message exchanges. Since the access times as well as transmission times can vary because of channel errors and retransmissions, the communication delay may be different for each direction. This is an important point for synchronization since most solutions rely on consecutive message exchanges where the round trip time between two nodes is considered for calculations. Whereas for wired networks the round-trip time is roughly equal to twice the delay in one direction, wireless communication results in asymmetric delays in each direction.

Additional Constraints

Besides energy limitations, low-power and low-cost sensor nodes are often constrained in their processor speeds and memory, further requiring that time synchronization protocols are lightweight. The small size and cost of sensor devices proscribe the use of large and expensive hardware to achieve synchronization (e.g., GPS receivers). Therefore, time synchronization protocols should be designed to operate in resource-constrained environments with little or no addition to the overall cost of a sensor device. Wireless sensor network deployments are often very large in scale and a synchronization protocol should scale well with increasing numbers of nodes or network density. Finally, different sensor applications will have differing requirements on clock accuracy or precision.

11.2 Basics of Time Synchronization

In this section, the fundamental aspects regarding the time synchronization process are covered. These aspects are the basis of the time synchronization protocols that will be examined in the next section. In these protocols, one node, called the *receiver*, exchanges data packets with another node, called the *sender*, to let the receiver synchronize to the sender's clock. Since synchronization can only be achieved through communication between nodes, the effects of the wireless channel need to be carefully considered

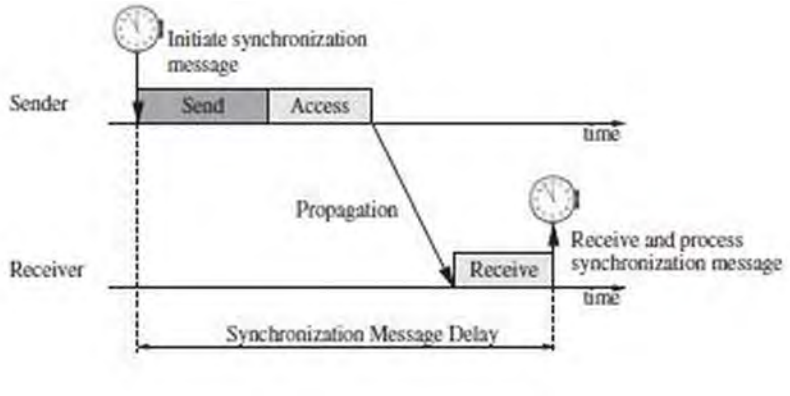


Figure 11.3 End-to-end delay for a synchronization message.

in the design of synchronization protocols. However, wireless communication introduces randomness in the delay between two nodes. Together with the several challenges that a time synchronization protocol has to face, the non-determinism of communication delay significantly contributes to the precision that can be achieved. In general, this latency experienced by synchronization messages is the sum of several components, as illustrated in Figure 11.3. Considering the handshake scheme shown in Figure 11.3 the delay between two nodes has four components:

1. *Send delay:* This is the time spent by the sender to generate the synchronization message and pass the message to the network interface. This includes delays caused by operating system behavior system call interface, context switches), the network protocol stack, and the network device driver. The sending delay is non-deterministic because of the complex and time-varying interactions between each hardware and software component in the embedded system.

2. *Access delay:* This is the time spent by the sender to access the physical channel and is mostly determined by the medium access control (MAC) protocol in use. As mentioned in Chapter 4, depending on the MAC protocol, an additional delay is introduced in waiting for access to the channel. While this delay may be bounded in a TDMA protocol because of reserved slots, a CSMA-based protocol may introduce a significant amount of access delay if the channel is highly loaded. In either case, the access delay cannot be determined a priori.

3. *Propagation delay:* It refers to the actual time needed for the message to reach from the sender to the receiver. Once a node accesses the channel, the synchronization message is transmitted and it takes some amount of time for the packet to reach the intended receiver. While the propagation delay is negligible in communication through

air, underground and underwater environments introduce significant propagation delay, which is important for synchronization. Moreover, this delay is directly proportional to the distance between the nodes.

4. *Receive delay:* This is the time spent by the receiver device to receive the message from the medium, to process the message, and to notify the host of its arrival. This includes the transmission and processing delay required for the antenna to receive the message from the channel, perform A/D conversion, and notify the operating system of its arrival. Host notification typically occurs via interrupts, at which the local time (i.e., the message arrival time) can be read. As a consequence, the receive delay tends to be much smaller than the send delay.

Synchronization is typically based on some sort of message exchange among sensor nodes. If the medium supports broadcast (as it is the case in wireless systems), multiple devices can be synchronized simultaneously with a low number of messages. Most existing time synchronization protocols are based on *pairwise synchronization*, where two nodes synchronize their clocks using at least one synchronization message. *Network-wide synchronization* can be achieved by repeating this process among multiple node pairs until every node in a network has been able to adjust its clock. Regarding the flow of synchronization messages, we can distinguish between three different techniques used to achieve synchronization:

11.2.1 One-Way Message Exchange

This is the simplest approach of pairwise synchronization. It occurs when only a single message is used to synchronize two nodes, that is, one node sends a time stamp to another node. As illustrated in Figure 11.4, node i sends a synchronization message to node j at time t_1, embedding t_1 as *time stamp* into the message. Upon reception of this message, node j obtains a time stamp t_2 from its own local clock. The difference between the two time stamps is an indicator of the clock offset (between the clocks of nodes i and j) δ. More accurately, the difference between the two times is expressed as

$$t_2 - t_1 = D + \delta\,, \tag{11.5}$$

where D is the unknown propagation time. Propagation times in the wireless medium are very small (a few microseconds) and are often ignored or assumed to be a certain constant value. In this approach, node j is able to calculate an offset and adjust its clock to match the clock of node i.

11.2.2 Two-Way Message Exchange

A somewhat more accurate approach is to use two synchronization messages as shown in Figure 11.5. Here, node j responds with a message issued

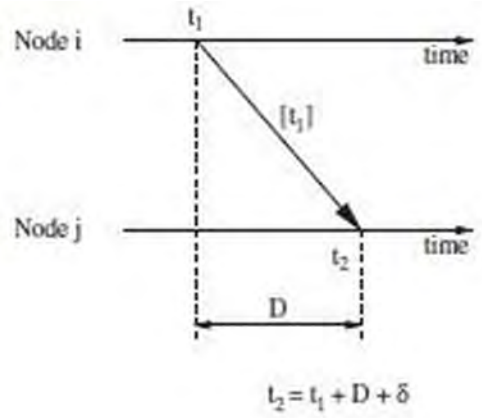


Figure 11.4 One-way message exchange procedure.

at time t_3, containing time stamps t_1, t_2, and t_3. Upon reception of this second message at time t_4, both nodes are able to determine the clock offset, again assuming a fixed value for the propagation delay. However, node i is now able to more accurately determine both the propagation delay and the offset as

$$D = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}, \tag{11.6}$$

$$\text{offset} = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}. \tag{11.7}$$

Note that this assumes that the propagation delay is identical in both directions and the clock drift does not change between measurements (which is feasible because of the brief time span). While only node i has sufficient information to determine the offset, node i can share the offset value with node j in a third message.

11.2.3 Receiver-Receiver Synchronization

A different approach is taken by protocols that apply the receiver-receiver synchronization principle, where synchronization is based on the time at which the same message arrives at each receiver. This is in contrast to the more traditional sender-receiver approach of most synchronization schemes. In broadcast environments, these receivers obtain the message at about the same time and then exchange their arrival times to compute an offset (i.e., the difference in reception times indicates the offset of their clocks). Figure 11.6 shows an example of this scheme where there are two node receivers j and k and three messages are needed to synchronize them. Note that the broadcast

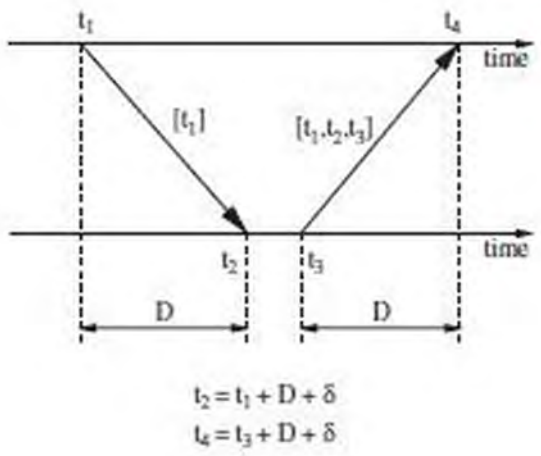


Figure 11.5 Two-way message exchange procedure.

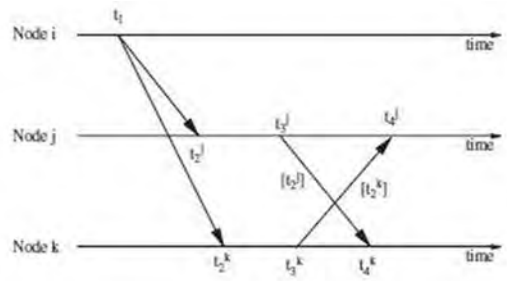


Figure 11.6 Receiver-receiver synchronization procedure.

message does not carry a time stamp, instead the arrival times of the broadcast message at the different receivers is used to synchronize the receivers to each other.

11.3 Time Synchronization Protocols

Numerous time synchronization protocols for WSNs have been developed, where most of them are based on some variations of the message exchange concepts described in the previous section.

Typical clock synchronization algorithms rely on the ability to exchange messages at a high rate which may not be possible in wireless sensor networks. Traditional time synchronization algorithms such as the *Network Time Protocol (NTP)*, which is a synchronization protocol used for the Internet, is not suitable for WSNs applications. However, this protocol contains several synchronization approaches that have been adopted by the synchronization protocols developed for WSNs. Before presenting the most representative

ones, the time synchronization method based on the MMSE criterion is provided in the following section.

11.3.1 MMSE Technique in Time Synchronization Protocols

In this method, a single node i with software clock $C_i(t)$ broadcasts a synchronization message. The objective is to establish the relative time among different clocks, while allowing the individual clocks to run freely. We also consider another node j that is unsynchronized with node i and receives the synchronization message. Moreover, the local clock of a node j can be represented relative to a node i as follows:

$$c_j(t) = a_0 + a_1 \cdot c_i(t) \,, \tag{11.8}$$

where a_0 and a_1 are the relative clock offset and drift, respectively. The synchronization between two nodes i and j is performed according to the relation between their clocks and the aim is to correct the relative differences between them. Consequently, providing a common reference frame requires that the reference clock drift and offset between the nodes in the network must be minimized by performing a linear least squares fit (linear regression). Since errors occur due to the non-determinism of the communication delay, as explained before, a sequence of broadcast messages in different time instants are sent and the values of $C_j(t)$ are periodically measured. Denote x_0 the value of the software clock of node i at the time instant t_0; that is $C_i(t_0) = x_0$. Then, the value of the software clock of node j becomes:

$$C_j(t_0) = a_0 + a_1 \cdot C_i(t_0) = a_0 + a_1 \cdot x_0 \stackrel{\Delta}{=} y_0 \,. \tag{11.9}$$

Continuing iteratively n times and assuming that a_0 and a_1 are constant we get

$$\begin{aligned} &C_i(t_1) \stackrel{\Delta}{=} x_1 \\ &C_j(t_1) = a_0 + a_1 \cdot x_1 \stackrel{\Delta}{=} y_1 \\ &\vdots \\ &C_i(t_{n-1}) \stackrel{\Delta}{=} x_{n-1} \\ &C_j(t_{n-1}) = a_0 + a_1 \cdot x_{n-1} \stackrel{\Delta}{=} y_{n-1} \,. \end{aligned} \tag{11.10}$$

Therefore, to find the best fit to the data according to the MMSE criterion, we form and solve

$$A \cdot X = Y \,, \tag{11.11}$$

where

$$A = \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \vdots & \vdots \\ x_{n-1} & 1 \end{bmatrix} \tag{11.12}$$

$$X = \begin{bmatrix} a_1 \\ a_0 \end{bmatrix}, \tag{11.13}$$

and

$$Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}. \tag{11.14}$$

The solution of this equation provides an estimator $\widehat{X} = L \cdot Y$ of the unknown vector X, where

$$L = \left(A^T \cdot A\right)^{-1} A^T,$$

and

$$A^T \cdot A = \begin{bmatrix} \sum_{i=0}^{n-1} {x_i}^2 & \sum_{i=0}^{n-1} x_i \\ \sum_{i=0}^{n-1} x_i & n \end{bmatrix},$$

$$A^T \cdot Y = \begin{bmatrix} \sum_{i=0}^{n-1} x_i y_i \\ \sum_{i=0}^{n-1} y_i \end{bmatrix}. \tag{11.15}$$

The following sections provide an overview of some representative time synchronization schemes and protocols.

11.3.2 The Network Time Protocol

The Network Time Protocol (NTP) is widely used in the Internet to establish synchronism among a very large number of devices. Its main characteristics are the scalability, the self-configuration over multiple hops, and the robustness to failures. In this scheme, synchronization among nodes is accomplished through a hierarchical structure of time servers each rooted to some source of external time (e.g., GPS, which, in turn, is rooted to coordinated universal time (UTC)), as shown in Figure 11.7. NTP relies on a two-way handshake (request/response messages) between two nodes in order to estimate delays between these nodes in the network and to adjust the local timing to the network standard. These delays tend to be dominated by congestion, which is highly variable. To minimize these effects, a sequence of messages is sent to multiple peers, with the shortest delay returns in each probing sequence used for delay and offset measurements for that peer (remote server). The results are then combined and clustered to use only the most reliable (consistent) of the peers, with a weighted combination of these results used as inputs to a phase/frequency lock loop that compares to NTP

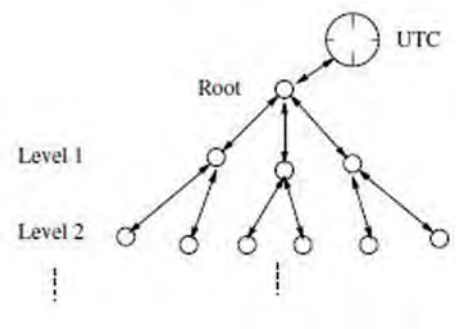


Figure 11.7 The hierarchical time server architecture of the NTP protocol.

time. This selection process is designed to reject unreliable or malicious peers. While NTP provides robust time synchronization in a large-scale network, many characteristics of WSNs make this protocol unsuitable. In particular, synchronizing all sensor nodes into a single clock reference may be a problem due to interference from the environment and the large variations in delays between the different parts of the sensor field. The interference can temporarily disjoint the sensor field into multiple smaller fields leading to undisciplined clocks among these smaller fields. Moreover, NTP is computationally intensive and requires a precise time server to synchronize the nodes in the network. In addition, it does not take into account the energy consumption required for time synchronization. Although NTP is robust, it may suffer from large propagation delays when sending timing messages to the time servers. In addition, the nodes are synchronized in a hierarchical manner and some of the time servers in the middle of the hierarchy may fail causing unsynchronized nodes in the network. NTP also assumes symmetric link delays between two nodes which may not be true in the case of WSNs.

11.3.3 Timing-Sync Protocol for Sensor Networks

The Timing-Sync Protocol for Sensor Networks (TPSN) adopts some concepts from NTP. Similar to NTP, a hierarchical structure is used to synchronize the whole WSN to a single time server. It is a sender-receiver synchronization approach aiming to provide network-wide time synchronization. The synchronization of a node depends on its parent in the hierarchical structure. Therefore, even if the number of nodes in the network increases, the high synchronization accuracy can still be achieved. Since the hierarchical structure covers the entire network based on a root node, the whole network can be synchronized to the same time reference.

More specifically, the TPSN algorithm elects a root node and builds a spanning tree (hierarchical structure) of the network during the initial level discovery phase. The root node initiates this phase by broadcasting a

level_discovery message that contains the level and the unique identity of the sender. Every immediate neighbour of the root node uses this message to identify its own level and rebroadcasts it with its own identity and level. This process is repeated until every node in the network has identified its level.

In the synchronization phase of the algorithm, nodes synchronize to their parent in the tree (a node from a higher level) by a two-way message exchange. Using the timestamps embedded in the synchronization messages, the child node is able to calculate the transmission delay and the relative clock offset. In specific, the synchronization phase is initiated by the root node issuing a *time_sync* packet. After waiting for some random time (to reduce contention during medium access), nodes in level 1 initiate the two-way message exchange with the root node. Once a node in level 1 receives an acknowledgement from the root, it computes its offset and adjusts its clock. Nodes on level 2 will overhear the synchronization pulses issued by their level 1 neighbors and after a certain backoff time they initiate their pairwise synchronization with nodes in level 1. The backoff time is necessary to give level 1 nodes time to receive and process the acknowledgement of their own synchronization pulses. This process is continued throughout the hierarchical structure until all nodes have synchronized to the root node.

However, TPSN does not compensate for clock drift which makes frequent resynchronization mandatory. In addition, TPSN causes a high communication overhead since a two-way message exchange is required for each child node. Since the design of TPSN is based on a hierarchical methodology similar to NTP, nodes within the hierarchy may fail and may cause nodes to become unsynchronized. Further, node mobility may render the hierarchy useless, because nodes may move out of their levels.

11.3.4 Lightweight Tree-Based Synchronization

Similar to TPSN described in the previous section, Lightweight Tree-Based Synchronization (LTS) relies on a tree structure in order to perform network-wide synchronization. The protocol is based on a message exchange between two nodes to estimate the clock drift between their local clocks. This pairwise synchronization scheme is extended for multi-hop synchronization. LTS can be used with different algorithms for both centralized and decentralized multi-hop synchronization

The centralized multi-hop version of LTS is based on a single reference node that is the root of a spanning tree comprising all nodes of the network which are synchronized to the root. In this mechanism, synchronization accuracy decreases as the depth of the spanning tree increases. This is due to that the errors resulting from the pairwise synchronizations are additive and therefore increase along the branches of the tree as a function of the number of hops. Thus, in order to maximize the synchronization accuracy,

the depth of the tree should be minimized. After the tree is constructed, the root of the tree initiates pairwise synchronization with its children nodes and the synchronization is propagated along the tree to the leaf nodes.

The distributed multi-hop version of LTS does not require the construction of a spanning tree and the synchronization responsibility is moved from the reference node to the sensor nodes themselves. This provides event-based synchronization capabilities, where each node performs synchronization only when it has a packet to send. In this case, each node is informed about its distance to the reference node for synchronization and adjusts its synchronization rate accordingly. Since synchronization accuracy is inversely proportional to distance, nodes farther apart from the reference node perform synchronization more frequently

11.3.5 Flooding Time Synchronization Protocol

The Flooding Time Synchronization Protocol (FTSP) differs from other solutions in that it uses a single broadcast to establish synchronization points between sender and receivers while eliminating most sources of synchronization error. A root node is elected which periodically floods its current time stamp into the network forming an ad-hoc tree structure. MAC layer time-stamping reduces possible sources of uncertainty in the message delay. Each node uses a linear regression table to convert between the local hardware clock and the clock of the reference node. The root node is dynamically elected by the network based on the smallest node identifier. After initialization, a node waits for a few rounds and listens for synchronization beacons from other nodes. Each node sufficiently synchronized to the root node starts broadcasting its estimation of the global clock. If a node does not receive synchronization messages during a certain period, it will declare itself the new root node.

11.3.6 Reference Broadcast Synchronization protocol

The Reference Broadcast Synchronization (RBS) protocol exploits the broadcast nature of the physical channel to synchronize a set of receivers with one another. It aims to inimize the critical path in synchronization by eliminating the effect of the sender. Instead of synchronizing a sender with a receiver, RBS provides time synchronization among a set of receivers that are within the reference broadcast of a sender. Since the propagation times are negligible, once a packet is transmitted by a sender, it is received at its neighbors almost at the same instant (Figure 11.8). Each of the receivers, which are within the broadcast range, records the time of arrival of the reference packets. Then, the receivers communicate with each other to determine the relative clock offsets. RBS is designed for single-hop time synchronization only. To provide multi-hop synchronization, RBS

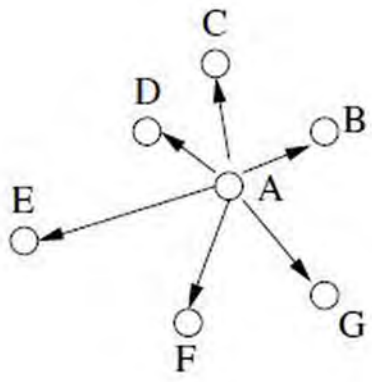


Figure 11.8 Reference broadcasting, where node A's broadcast message is used by the remaining nodes for synchronization.

uses nodes that are receiving two or more reference broadcasts from different transmitters. These nodes are denoted as translation nodes and are used to translate the time between different broadcast domains.

11.3.7 Time-Diffusion Synchronization Protocol

Both the RBS and TPSN protocols aim to provide multi-hop synchronization by extending a single hop synchronization method and performing this method iteratively. While these solutions may be acceptable for small networks, they are not scalable for very large networks. Instead, the **Time-Diffusion Synchronization Protocol (TDP)** aims to maintain a common time throughout the network within a certain tolerance. This tolerance level can be adjusted depending on the application and the requirements of the network. TDP assigns three different duties to the nodes in the network to provide multi-hop synchronization: master nodes, which initiate the synchronization process, diffused leader nodes that propagate the synchronization messages further away from the broadcast range of the master nodes and regular nodes that participate in the synchronization process minimally.

TDP follows a periodic procedure consisting of active and passive phases. Resembling a TDMA-like operation, the synchronization procedures are performed during the active phase. The protocol then enters the passive phase, where no timing updates are performed. As the duration of the passive phase increases, the network deviates further from the equilibrium time, which necessitates resynchronization. On the other hand, a smaller passive period results in more frequent synchronization, increasing the overhead of the protocol. TDP provides an adaptive mechanism to adjust the synchronization schedule according to the timing requirements of the WSN.

The active phase of TDP is further divided into cycles of length τ. The

master nodes are re-elected at each cycle and the synchronization operation is performed. Each cycle of length τ is further divided into rounds of length δ, where the master nodes repeatedly broadcast synchronization messages. According to this structure, TDP consists of two major procedures: the election/re-election procedure (ERP), where the master nodes and the diffused leaders are selected; and the time diffusion procedure (TP), where the synchronization is performed. The ERP is performed at the beginning of each cycle, τ, while the TP is performed at each round, δ, following the ERP. The ERP aims to differentiate nodes that are eligible to become master nodes or diffused leaders and performs this selection. The re-election mechanism, which is performed every τ seconds, helps to distribute the load on the master nodes and the diffused leaders.

TDP supports node mobility. It assigns master and diffused leader duties to nodes dynamically based on their clock quality as well as the remaining energy. As a result, the energy consumption is also equally distributed in the network. In addition, TDP does not require an external time server for synchronization. Without time servers, the protocol can reach a network-wide equilibrium time, which can be converted into another frame of reference by interacting with a single node in the network. However, the election/re-election procedure, which dynamically elects master and diffused leader nodes, increases the complexity of the protocol because of multiple periods and cycles. If done too frequently, the energy consumption of the protocol also increases.

11.3.8 Mini-Sync and Tiny-Sync

Tiny-sync and mini-sync protocols have been developed to provide a simple and accurate time synchronization for WSNs. Both protocols are based on a hierarchical structure of sensor nodes, where each node is synchronized with its parent node. In tiny- and mini-sync protocols, each node estimates the relative clock drift and offset to its parent node in the hierarchical tree structure for synchronization.

More specifically, recall that the local clock of a node i can be represented relative to a node j as follows:

$$C_i(t) = a_{ij}C_j(t) + b_{ij}\,, \tag{11.16}$$

where a_{ij} expresses the relative drift and b_{ij} the relative offset, respectively. In order to determine the unknown elements in the above equation, nodes (nodes 1 and 2) use the two-way messaging handshake, for example, node 1 sends a time-stamped probe message at time t_0 to node 2 and node 2 responds immediately with a time-stamped reply message at time t_1. Node 1 records the arrival time of the second message (t_2) to obtain a 3-tuple of time stamps (t_0, t_1, t_2), which is called a data point. Since t_0 happened

before $t-1$, and t_1 happened before t_2, the following inequalities should hold:

$$t_0 < a_{12}t_1 + b_{12}\,, \tag{11.17}$$

$$t_2 > a_{12}t_1 + b_{12}\,. \tag{11.18}$$

This procedure is repeated multiple times, resulting in a series of data points and new constraints on the admissible values of a_{12} and b_{12}. As the node gathers more data points, the accuracy of the estimation improves. However, since sensor nodes are constrained in terms of memory, this estimation should be performed with a minimum number of data points.

The two versions of the protocol are based on the observation that not all data points are useful. Every data point results in two constraints for the relative drift and offset. The Tiny-sync algorithm maintains only four of these constraints, that is, whenever a new data point has been obtained, the current four and the two new constraints are compared and only the four constraints that result in the best estimates of offset and drift are kept. While this operation limits the number of constraints and the computational complexity, it does not always lead to the optimum result. More specifically, the data points discarded in an earlier round may be more helpful for determining closer bounds in the following rounds. As a result, tiny-sync provides a lightweight operation at the cost of deviating from the optimum clock drift and offset values.

Therefore, tiny-sync is extended through the mini-sync protocol which stores more data points to determine the optimum values for the relative clock drift and offset. Instead of selecting four data points for each three measurements and discarding the other two, mini-sync discards the data points only if they are proven not to improve the estimate. This results in larger computational and storage costs compared to Tiny-sync, but the advantage is an increased precision.

11.4 The Gradient Time Synchronization Protocol

The Gradient Time Synchronization Protocol (GTSP) is a representative clock synchronization protocol that performs synchronization between nodes using an exclusively distributed scheme. It is inspired by a long list of theoretical papers, originating in the distributed computing community, lately also being adopted by the control theory community. It relies only on local information, requiring no reference node or tree construction. Figure 11.9 shows the comparison between the FTSP and GTSP schemes. As mentioned before, FTSP and similar protocols work on a spanning tree, synchronizing nodes in the tree with their parents, and ultimately with the root of the tree.

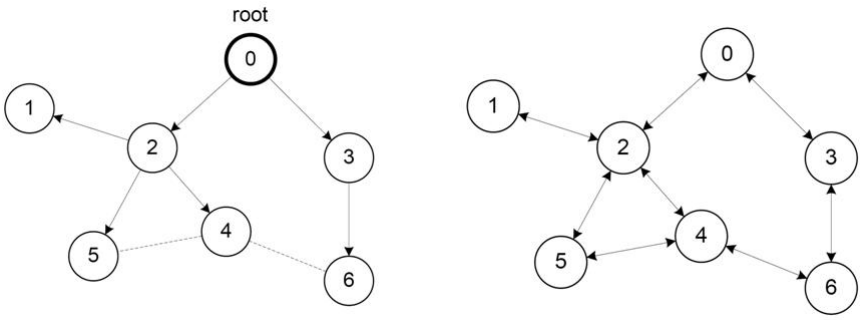


Figure 11.9 FTSP vs GTSP protocols comparison.

In this case, neighbouring nodes which are not closely related in the tree, i.e., where the closest common ancestor even is the root of the tree, will not be synchronized well because errors propagate down differently on different paths of the tree. On the other hand, the GTSP approach provides precise clock synchronization between direct neighbours while still maintaining a tolerable global skew.

Before discussing the basic principles of GTSP scheme, we assume a network consisting of a number of nodes equipped with a hardware clock subject to clock drift. Therefore, each sensor node i is equipped with a hardware clock $H_i(\cdot)$ whose value at time t is defined as

$$H_i(t) = \int_{t_0}^{t} h_i(\tau)\, d\tau + \phi_i(t_0) \ , \tag{11.19}$$

where $h_i(\tau)$ is the hardware clock rate at time τ and $\phi_i(t_0)$ is the hardware clock offset at time t_0. Moreover, the software clock of node i, $C_i(\cdot)$, is computed as a function of the current hardware clock with its value representing the synchronized time of node i. It is calculated as follows:

$$c_i(t) = \int_{t_0}^{t} h_i(\tau)\, l_i(\tau)\, d\tau + \theta_i(t_0) \ , \tag{11.20}$$

where $l_i(\tau)$ is the relative logical clock rate and $\theta_i(t_0)$ is the clock offset between the hardware clock and the logical clock at the reference time t_0. Defining the absolute logical clock rate $x_i(t)$ of node i at time t, we have

$$x_i(t) \triangleq h_i(t) \cdot l_i(t) \ , \tag{11.21}$$

the goal is to have the same $x_i(t)$ for all nodes taking into account the possible network topology changes; that means that in different time periods, the neighbour nodes of node i may differ.

The synchronization process begins with every node i periodically broadcasting a synchronization beacon containing its current logical time $C_i(t)$ and the relative logical clock rate $l_i(t)$. Having received beacons from all neighbouring nodes during a synchronization period, node i uses this information to update its absolute logical clock rate $x_i(t)$ according to the following rule:

$$x_i\left(k+1\right) \triangleq \frac{\sum\limits_{j \in N_i(k)} x_j\left(k\right) + x_i\left(k\right)}{|N_i|+1}\,, \tag{11.22}$$

where $N_i(t_k)$ is the set of neighbors of node i at time t_k, and $|N_i|$ denotes the cardinality of such a set. The aim is to show that using this update mechanism all nodes converge to a common logical clock rate x_{ss}, which means we wish to show

$$\lim_{t\to\infty} x_i(t) = \lim_{t\to\infty} h_i(t)\cdot l_i(t) = x_{\text{ss}}, \quad \forall i\,. \tag{11.23}$$

By putting in a vector the logical clocks rates of all the N nodes, we have

$$\mathbf{X}\left(k+1\right) = A\left(k\right)\cdot\mathbf{X}\left(k\right)\,, \tag{11.24}$$

where the vector $\mathbf{X}(k) = (x_1(k), x_2(k), \ldots, x_n(k))^T$ contains the logical clock rates of the N nodes at time k. The entries of the $n\times n$ matrix A are defined as

$$A\left(k\right) = \left[a_{i,j}\left(k\right)\right] = \begin{cases} \frac{1}{|N_i|+1} & \text{if } i,j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{11.25}$$

Matrix A has the row stochasticity property where:

$$A\left(k\right)\cdot\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{11.26}$$

Supposing that the graph $G(V,E)$, which represents the network and corresponds to matrices A, is connected in the long run then all the logical clock rates will converge to a steady-state value x_{ss}:

$$\lim_{k\to\infty} X\left(k\right) = x_{\text{ss}}\cdot\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \tag{11.27}$$

and therefore synchronization is achieved.

Problems

Problem 11.1 TOA with low-cost clocks (Ex.9.4 in (Pottie and Kaiser, 2005))

In order to make accurate range measurements in a GPS system, the receiver and satellite both need clocks that can be synchronized down to the nanosecond, which potentially could require atomic clocks not only on all the satellites, but also in the receivers. However, atomic clocks are far too expensive for everyday consumer use. GPS sidesteps this problem by measuring the distance to four instead of the minimum three located satellites. Every satellite contains an expensive atomic clock, but the receiver uses an ordinary quartz clock, which it constantly resets. With four range measurements, the receiver can easily calculate the necessary adjustment that will cause the four spheres to intersect at one point. Based on this, it resets its clock to be in sync with the satellite's atomic clock, thus providing time as well as location. Explain mathematically how this fourth measurement provides these benefits.

Problem 11.2 Time difference of arrival (TDOA) in a two-dimensional space (Ex.9.5 in (Pottie and Kaiser, 2005))

TOA requires that all the reference nodes and the receiver have precise synchronized clocks and the transmitted signals be labeled with time stamps. TDOA measurements remove the requirement of an accurate clock at the receiver. Assume that five reference nodes have known positions $(0,0), (-1,-1), (0,1), (3,1)$, and $(1,4)$ respectively. We choose $(0,0)$ as the reference sensor for differential time-delays which are defined as

$$t_{1r} = t_1 - t_r = \frac{r_{s1} - r_{s2}}{v},$$

where v is the velocity of propagation, r_{si} is the distance between the unknown node and the ith node. Further assume that $t_{12} = -1.4s$, $t_{13} = 0.4s$, $t_{14} = -1.6s$, and $t_{15} = -2.6s$.

(a) Find the unknown location (x_t, y_t).

(b) Now assume that the propagation speed is known as 1.8 m/s. Find the unknown location (x_t, y_t).

Problem 11.3 TDOA in a three-dimensional space (Ex.9.6 in (Pottie and Kaiser, 2005))

Now assume that five reference nodes are known at $(0,3,0)$, $(6,0,0)$, $(3,4,0)$, $(-4,-3,0)$, and $(0,0,-8)$ respectively. Also, $t_{12} = 0s$, $t_{13} = 1s$, $t_{14} = 0.7s$, $t_{15} = 0.7s$, and $t_{16} = 1.7s$. The velocity of propagation is v.

(a) Find the unknown location (x_t, y_t, z_t) using (9.10) from lecture notes.

(b) Now assume that the propagation speed is known to be 8.7 m/s. Find the unknown location (x_t, y_t, z_t) using (9.12) from lecture notes.

Problem 11.4 Ex.9.3 in (Dargie and Poellabauer, 2010)

Consider two nodes, where the current time at node A is 1100 and the current time at node B is 1000. Node A's clock progresses by 1.01 time units once every 1 s and node B's clock progresses by 0.99 time units once every 1 s. Explain the terms clock offset, clock rate, and clock skew using this concrete example. Are these clocks fast or slow and why?

Problem 11.5 Ex.9.4 in (Dargie and Poellabauer, 2010)

Assume that two nodes have a maximum drift rate from the real time of 100 ppm each. Your goal is to synchronize their clocks such that their relative offset does not exceed 1 s. What is the necessary re-synchronization interval?

Problem 11.6 Ex.9.6 in (Dargie and Poellabauer, 2010)

A network of five nodes is synchronized to an external reference time with maximum errors of 1, 3, 4, 1, and 2 time units, respectively. What is the maximum precision that can be obtained in this network?

Problem 11.7 Ex.9.7 in (Dargie and Poellabauer, 2010)

Node A sends a synchronization request to node B at 3150 (on node A's clock). At 3250, node A receives the reply from node B with a times-tamp of 3120.

(a) What is node A's clock offset with respect to the time at node B (you can ignore any processing delays at either node)?

(b) Is node A's clock going too slow or too fast?

(c) How should node A adjust the clock?

Problem 11.8 Ex.9.8 in (Dargie and Poellabauer, 2010)

Node A issues a synchronization request simultaneously to nodes B, C, and D(Figure 11.10). Assume that nodes B, C, and D are all perfectly synchronized to each other. Explain why the offsets between node A and the three other nodes may still differ?

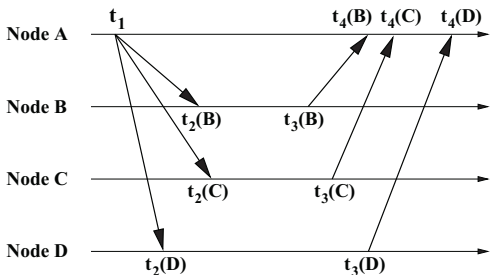


Figure 11.10 Pairwise synchronization with multiple neighboring nodes.

Chapter 12

Wireless Sensor Network Control Systems

The concept of control in WSNs is of major importance. As we have learnt in the previous chapters of the book, WSNs provide extensive information from the physical world through distributed sensing mechanisms. With the emergence of low-cost controllers that can affect the environment, information that is sensed from the environment can be utilized to act on the environment. This led to the development of Wireless Sensor Network Control Systems (WSN-CS) that are capable of observing the physical world, processing the data, making decisions based on the sensor observations and performing appropriate actions. This chapter is devoted to the study of such systems.

The chapter is organised as follows: Section 12.1 covers some basic notions from control theory, as well as a few useful mathematical results concerning stability. In section 12.2, the general architecture of the WSN-CS is described and equations for a sampled system are derived. Section 12.3 discusses the challenges for networked control systems when it comes to stability. Criteria for stability are also given. Section 12.4 explores different kinds of methods for sampling the plant output of the WSN-CS. In section 12.5, different approaches for designing a WSN-CS are evaluated. In the last two sections, we study two types of systems that are rather different from the ones studied in the first sections. Specifically, section 12.6 deals with a Model-Based Network Control System and section 12.7 covers a feedback control system in which every node of a wireless network acts as part of the controller.

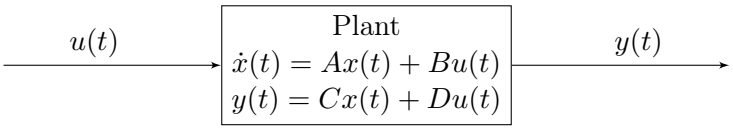


Figure 12.1 The state space description.

12.1 Preliminaries

12.1.1 State space representation

By a plant we mean a physical phenomenon or process that is controlled in order to make it behave in a desirable way. In order to describe the plant and its time-evolution, we shall use the **state space representation**. Hence, at every point in time t, the plant is assumed to be described by the **system state** $x(t) \in \mathbb{R}^n$. Furthermore, a plant is influenced by an input signal $u(t) \in \mathbb{R}^m$ and produces a measurable output signal $y(t) \in \mathbb{R}^p$. Unless otherwise stated, all plants that we will be concerned with are linear, time-invariant and operate in continuous-time. Hence, they are governed by the following equations.

$$\frac{dx}{dt}(t) := \dot{x}(t) = Ax(t) + Bu(t) \tag{12.1}$$

$$y(t) = Cx(t) + Du(t)\,, \tag{12.2}$$

where A, B, C and D are assumed to be known matrices of appropriate dimensions. The situation we have described is illustrated in Figure 12.3.

Theorem 12.1.1. *Given $x_0 = x(0)$, the unique solution of* (12.1) *is given by:*

$$x(t) = e^{A(t)}x(0) + \int_0^t e^{A(t-s)}Bu(s)ds \qquad t > 0\,. \tag{12.3}$$

Proof. A proof can be found in an introductory book on control theory and is omitted here. For a proof see e.g. (Glad and Ljung, 1981). □

In the following simple lemma, we will rewrite equation (12.3) in a way that will be useful later in the chapter.

Lemma 12.1.2. *Let $t_0 > 0$. Equation* (12.3) *may be written as:*

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^t e^{A(t-s)}Bu(s)ds \qquad t > t_0\,. \tag{12.4}$$

Proof. With $t = t_0$ in (12.3) we get

$$\begin{aligned} & e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-s)}Bu(s)ds \\ & = e^{A(t-t_0)}\left(e^{At_0}x(0) + \int_{0}^{t_0} e^{A(t_0-s)}Bu(s)ds\right) + \int_{t_0}^{t} e^{A(t-s)}Bu(s)ds \\ & = e^{At}x(0) + \int_{0}^{t} e^{A(t-s)}Bu(s)ds = x(t)\,. \end{aligned}$$

□

12.1.2 Stability of difference equations

Let a function $g : \mathbb{R}^n \to \mathbb{R}^n$ be given and consider the following difference equation:

$$x(k+1) = g(x(k))\,. \tag{12.5}$$

A specific solution $x^*(k)$ of (12.5) is called **stable** if for all $\varepsilon > 0$ there is a $\delta > 0$ such that for every other solution $x(k)$ we have:

$$\|x\,(0) - x^*\,(0)\| \leq \delta \Rightarrow \forall k \in \mathbb{N} : \|x(k) - x^*\,(k)\| \leq \varepsilon\,.$$

In words, this means that every solution $x(k)$ that starts near a particular stable solution $x^*(k)$ will continue to stay near $x^*(k)$ for all time(i.e. for all k). Note however that the stability of $x^*(k)$ does not guarantee that $x(k)$ will converge to $x^*(k)$. The following stronger notion of stability ensures this convergence.

A specific solution $x^*(k)$ of (12.5) is called **asymptotically stable** if it is stable and if there is a $\delta > 0$ such that for every other solution $x(k)$ it holds that:

$$\|x\,(0) - x^*\,(0)\| \leq \delta \Rightarrow \|x\,(k) - x^*\,(k)\| \to 0 \quad as\ k \to \infty\,.$$

Assume that $x^*(k) = 0$ is a solution of (12.5). Then Figure 12.2 illustrates the behaviour of solutions that start out near the origin in the case when x^* is stable and in the case when x^* is asymptotically stable.

Now we shall turn to the important special case that $g(x(k))$ is a linear function, i.e. it is given by some matrix $A \in R^{n\times n}$. The difference equation (12.5) then becomes

$$x(k+1) = Ax(k)\,. \tag{12.6}$$

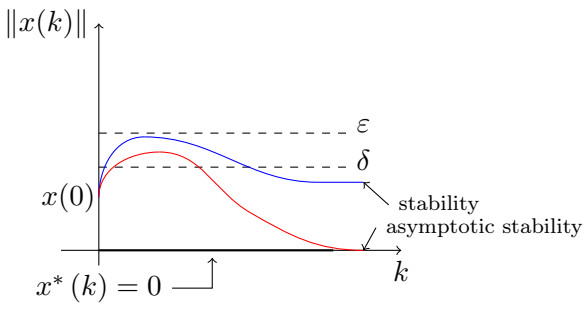


Figure 12.2 Stability vs asymptotic stability of the constant solution $x^*(k) = 0$.

The following lemma shows that for a linear difference such as (12.6), stability and asymptotic stability are properties of the difference equation rather than of a particular solution.

Lemma 12.1.3. *If the linear difference equation* (12.6) *has a (asymptotically) stable solution, then every solution of* (12.6) *is (asymptotically) stable.*

Proof. We give the proof for stability, the proof for asymptotic stability is completely analogous. Assume that $x^*(k)$ is a stable solution of (12.6). Let $x'(k)$ be another solution and consider some $\varepsilon > 0$. Pick $\delta > 0$ such that for every solution $x(k)$, it holds for all k that $\|x^*(k) - x(k)\| < \varepsilon$ provided that $\|x^*(0) - x(0)\| < \delta$. Now, in order to prove the stability of $x'(k)$, we take a solution $x''(k)$ such that $\|x'(0) - x''(0)\| < \delta$. We note that $x^*(k) + x'(k) - x''(k) = A^k x^*(0) + A^k x'(0) - A^k x''(0) = A^k(x^*(0) + x'(0) - x''(0))$, which shows that $x^*(k) + x'(k) - x''(k)$ is in fact also a solution. Furthermore, $\|(x^*(0) + x'(0) - x''(0)) - x^*(0)\| = \|x'(0) - x''(0)\| \leq \delta$ implies that $\|x'(k) - x''(k)\| = \|(x^*(k) + x'(k) - x''(k)) - x^*(k)\| \leq \varepsilon$ for all k. This shows that $x'(k)$ is a stable solution and the proof is finished. □

In light of Lemma 12.1.3 it makes sense to make the following definition. A linear difference equation of the form (12.6) is (asymptotically) stable if the constant solution $x^*(k) = 0$ is (asymptotically) stable. Thus a linear difference equation is stable if for all $\varepsilon > 0$ there is a $\delta > 0$ such that for every solution $x(k)$ we have

$$\|x(0)\| \leq \delta \Rightarrow \forall k : \|x(k)\| \leq \varepsilon .$$

Similarly, a system is asymptotically stable if it is stable and if there is a $\delta > 0$ such that for every solution $x(k)$ we have

$$\|x(0)\| \leq \delta \Rightarrow \|x(k)\| \to 0 \quad as \ k \to \infty .$$

As we shall see in the theorem 12.1.5, it turns out that the stability and asymptotic stability of a difference equation such as (12.6) is determined by the eigenvalues of A. Before we state the theorem we introduce the notation $\rho(A) = \max\{|\lambda|, \lambda \text{ is an eigenvalue of } A\}$. We also need the following lemma.

Lemma 12.1.4. *Let $D \in \mathbb{R}^{n\times n}$ be a diagonal matrix with diagonal entries $\lambda_1, \lambda_2, \ldots, \lambda_n$, such that $|\lambda_i| \leq 1$ for $i = 1, 2, \ldots, n$. Then $\|D\| \leq 1$.*

Proof. First let $e_i, e_2, \ldots, e_n$ denote the standard basis of $\mathbb{R}^n$. Now let $x \in \mathbb{R}^n$ have unit length and assume that $x = \sum_{i=0}^{n} c_i e_i$, where $c_1, c_2, \ldots, c_n \in \mathbb{R}$. Note that $\|x\|^2 = \sum_{i=1}^{n} c_i^2 = 1$. Thus

$$\|Dx\|^2 = \left\|\sum_{i=1}^{n} \lambda_i c_i e_i\right\|^2 = \sum_{i=1}^{n} \lambda_i^2 c_i^2 \leq \sum_{i=1}^{n} c_i^2 = 1 \Rightarrow \|Dx\| \leq 1\,.$$

Since x was an arbitrary unit-length vector it follows that $\|D\| \leq 1$. □

Theorem 12.1.5.

(i) The difference equation (12.6) *is stable if and only $\rho(A) \leq 1$.*

(ii) The difference equation (12.6) *is asymptotically stable if and only $\rho(A) < 1$.*

Proof. Throughout this proof we will assume that A is diagonalizable. Specifically, we assume that there exists an invertible matrix T and a diagonal matrix D such that $A = TDT^{-1}$.

(i) First assume that $\rho(A) \leq 1$. Let $\varepsilon > 0$ be given and pick

$$\delta = \frac{\varepsilon}{\|T\| \cdot \|T^{-1}\|}\,.$$

Let $x(k)$ be a solution of (12.6) such that $\|x(0)\| < \delta$. Then $\|x(k)\| = \|Ax(k-1)\| = \cdots = \left\|A^k x(0)\right\| = \left\|(TDT^{-1})^k x(0)\right\| = \left\|TD^k T^{-1} x(0)\right\| \leq \|T\| \left\|D^k\right\| \left\|T^{-1}\right\| \|x(0)\| \leq \|T\| \|D\|^k \left\|T^{-1}\right\| \|x(0)\|$.
Now since $\rho(A) \leq 1$ and since the diagonal entries of D are precisely the eigenvalues of A, we may apply Lemma 12.1.4 and conclude that $\|D\| \leq 1$. This implies that

$$\begin{aligned}\|x(k)\| &\leq \|T\| \|D\|^k \left\|T^{-1}\right\| \|x(0)\| \leq \|T\| \left\|T^{-1}\right\| \delta \\ &= \|T\| \left\|T^{-1}\right\| \frac{\varepsilon}{\|T\| \cdot \|T^{-1}\|} = \varepsilon\,.\end{aligned}$$

Thus the system is stable.

To prove the other direction, assume instead that $\rho(A) > 1$. Then there are $x' \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ such that $|\lambda| > 1$, $\|x'\| = 1$ and $Ax' = \lambda x'$. Now for any $\delta > 0$ we have a solution $x_\delta(k) = \delta A^k x'$. This solution satisfies $\|x_\delta(0)\| = \|\delta x'\| = \delta$, but

$$\|x_\delta(k)\| = \left\|A^k \delta x'\right\| = \delta|\lambda|^k \left\|x'\right\| = \delta|\lambda|^k \xrightarrow[k\to\infty]{} \infty\,.$$

This shows that (12.6) is not stable.

(ii) This time we assume first that $\rho(A) < 1$. Let $b_1, b_2, \ldots, b_n$ be a basis for $\mathbb{R}^n$ of eigenvectors of A and let $\lambda_1, \lambda_2, \ldots \lambda_n$ be the corresponding eigenvalues. Then

$$\|x(k)\| = \left\|A^k x(0)\right\| = \left\|\sum_{i=1}^{n} \lambda_i^k c_i b_i\right\| \leq \sum_{i=1}^{n} |\lambda_i|^k |c_i| \, \|b_i\| \xrightarrow[k\to\infty]{} 0\,.$$

We conclude that the system is asymptotically stable.

Now assume that $\rho(A) \geq 1$. Then there are $x' \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ such that $|\lambda| \geq 1$, $\|x'\| = 1$ and $Ax' = \lambda x'$. Again, for any $\delta > 0$ we have a solution $x_\delta(k) = \delta A^k x'$ that satisfies $\|x_\delta(0)\| = \|\delta x'\| = \delta$. Now we have $\|x_\delta(k)\| = \left\|A^k \delta x'\right\| = \delta|\lambda|^k \left\|x'\right\| \not\to 0$ as $k \to \infty$. Thus in this case the system is not asymptotically stable and the proof is finally completed.

□

12.2 The Wireless Sensor Network Control System

12.2.1 Definition

By a Wireless Sensor Network Control System(WSN-CS), we mean a spatially distributed feedback control system in which the sensor nodes of a WSN is used both to gather information from the plant and as a communication medium for the exchange of control signals. The plant information gathered by the sensors is sent over the wireless channel to a single dedicated controller. The goal of the controller is to bring the plant state into a desired region by sending a control decision to the actuators, which are the devices responsible for the execution of the control decisions.

We stress that in a WSN-CS, the system is controlled by a computer. Computers work in discrete time, i.e. they can only receive and send information at discrete points in time. Hence, the continuous output signal from the WSN-CS must be sampled before it is sent to the controller(c.f. appendix C). The discrete-time controller then decides a control signal which is

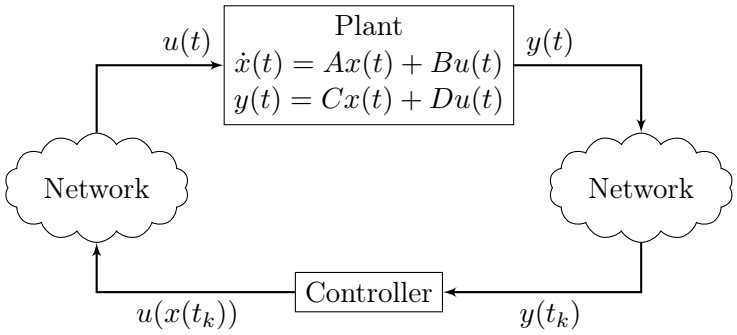


Figure 12.3 Model of a WSN-CS.

converted into a continuous signal before it is fed back to the plant through the actuators. This conversion is called signal reconstruction.

12.2.2 Model

We shall now describe the mathematical model of the WSN-CS that we will use. First of all, the WSN-CS have a plant with state $x(t) \in \mathbb{R}^n$, input signal $u(t) \in \mathbb{R}^m$ and output signal $y(t) \in \mathbb{R}^p$. It is governed by the following equations(c.f. section 12.1):

$$\frac{dx}{dt}(t) := \dot{x}(t) = Ax(t) + Bu(t) \tag{12.7}$$

$$y(t) = Cx(t) + Du(t)\,. \tag{12.8}$$

We denote the **sampling instants**, i.e. the times when the output signal $y(t)$ are sampled, by t_k and the **sampling period** $h(k)$ is defined by $h(k) = t_{k+1} - t_k$. We assume that we have a state feedback control system. Thus, when a sample $y(t_k)$ of the plant output arrives at the controller, the controller estimates the plant state $x(t_k)$ and decides a control signal $u(x(t_k))$, which is used as input for the plant. Unless otherwise stated, this estimation is assumed to be perfect, i.e. we assume that the controller has perfect knowledge of $x(t_k)$. We shall also assume that the signal reconstruction method is zero-order hold and hence

$$u(t) = u(t_k), \quad t \in [t_k, t_{k+1})\,. \tag{12.9}$$

For the rest of the chapter we shall, unless otherwise stated, assume that the sampling period $h(k)$ is constant and that $t_0 = 0$. It follows that $t_k = kh$. From (12.9) we then have that $u(t) = u(kh)$ for $t \in [kh, kh + h)$. The model we have described is illustrated in Figure 12.3.

Proposition 12.2.1. *The system state $x(t)$ of the WSN-CS we have described above satisfies:*

$$x(kh+h) = \phi x(kh) + \Gamma u(kh) \tag{12.10}$$

where $\phi = e^{Ah}$ *and* $\Gamma = \int_0^h e^{As} ds B$.

Proof. Equation (12.4) yields:

$$\begin{aligned} x(kh+h) &= e^{Ah}x(kh) + \int_{kh}^{kh+h} e^{A(kh+h-s)}Bu(s)ds \\ &= \phi x(kh) + \int_{kh}^{kh+h} e^{A(kh+h-s)}dsBu(kh) = \{s' = kh+h-s\} \\ &= \phi x(kh) - \int_{h}^{0} e^{As'}ds'Bu(kh) \\ &= \phi x(kh) + \int_{0}^{h} e^{As'}ds'Bu(kh) = \phi x(kh) + \Gamma u(kh)\,. \end{aligned}$$

□

Equation (12.10) is referred to as the **system equation for the sampled system**. Note that while proposition 12.2.1 gives a convenient description of the sampled system at the sampling instants, it provides no information about the system state in the intervals between these instants. The corollary below establishes an explicit solution of the system equation for the sampled system.

Corollary 12.2.2. *At the sampling instans $t_k = kh$, the system state of a WSN-CS is given by*

$$x(kh) = \phi^k x(0) + \sum_{j=0}^{k-1} \phi^{k-j-1}\Gamma u(jh)\,. \tag{12.11}$$

Proof. The proof is by induction over k. The base case $k = 0$ is trivial.

Assume that (12.11) holds for a certain k and consider $k+1$:

$$\begin{aligned} x(kh+h) &= \phi x(kh) + \Gamma u(kh) = \phi\left(\phi^k x(0) + \sum_{j=0}^{k-1} \phi^{k-j-1}\Gamma u(jh)\right) + \Gamma u(kh) \\ &= \phi^{k+1}x(0) + \sum_{j=0}^{k-1} \phi^{k-j}\Gamma u(jh) + \Gamma u(kh) \\ &= \phi^{k+1}x(0) + \sum_{j=0}^{k} \phi^{k-j}\Gamma u(jh)\,. \end{aligned}$$

Hence (12.11) is satisfied for $k+1$ and we are done. □

Example 12.2.3. *Consider a WSN-CS with a discrete-time controller that outputs a control signal that is a linear function of the state, i.e.* $u(kh) = -Kx(kh)$ *for some* $K \in \mathbb{R}^{n\times n}$*. The system state at the sampling instances is given by* (12.11)*:*

$$x(kh) = \phi^k x(0) - \sum_{j=0}^{k-1} \phi^{k-j-1}\Gamma K x(jh)\,.$$

12.3 Challanges for system stability

We shall say that a WSN-CS is (asymptotically) stable if the system equation for the sampled system is stable. For a WSN-CS, the presence of a wireless network in the control loop poses many challenges for the stability. In this chapter we shall deal with three categories of such challenges, namely the following:

(i) **Network delay** is the delay that occurs while data is exchanged between devices connected to the shared medium.

(ii) **Packet losses** may occur during packet transmission due to the unreliability of the transmission channel.

(iii) **Multiple-packet transmission** must sometimes be used due to bandwidth and packet size constraints.

12.3.1 Network delay

In the network there are two sources of delay, namely sensor-to-controller delay and controller-to-actuator delay. We denote these by τ_{sc} and τ_{ca} respectively(see fig. 12.4). If the controller is time-invariant, we may lump these terms together into a total network delay $\tau = \tau_{cs} + \tau_{sa}$. Furthermore,

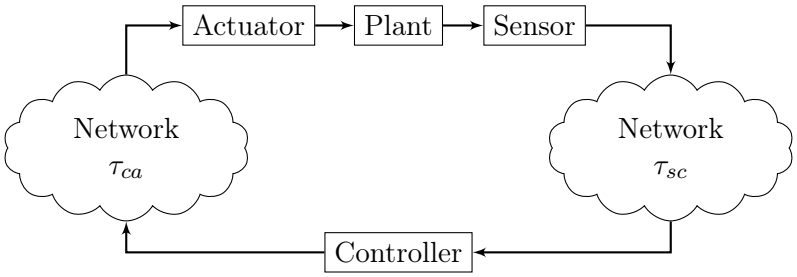


Figure 12.4 WSN-CS with packet losses and negligible delay.

we may also include any controller computational delay into τ. We can convince ourself that this simplification is valid by the following argument: If the controller is time-invariant, then the controller decision $u(t)$ will not depend on the time at which the sample $x(kh)$ reaches the controller and thus only the total delay is of importance for the system. In what follows we will study the stability of WSN-CSs which are subject to network delay. First we will consider a delayed system with a continuous-time plant and show how the system stability can be determined in the case of a constant network delay. Then we will consider a discrete-time plant subject to time varying delay and present a stability criterion for such a system. We will end this subsection with some comments on the relationship between the sampling interval, the network delay and the system stability.

Delayed system with continuous plant

The setting will be as follows. We have a WSN-CS that is subject to a total time-delay τ_k at time $t = kh$. Furthermore we assume that $\tau_k < h$ for all $k \in \mathbb{N}$. Mathematically, we may model this situation by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t - \tau_k), \quad t \in [kh, kh + h) \\ y(t) &= Cx(t) + Du(t)\,, \end{aligned}$$

where $u(t)$ is the input signal in the abscence of delay, i.e. $u(t) = u(kh)$ for $t \in [kh, kh + h)$.

Proposition 12.3.1. *The system state of the WSN-CS with delay as described above satisfies the following difference equation:*

$$x(kh + h) = \phi x(kh) + \Gamma_0(\tau_k)u(kh) + \Gamma_1(\tau_k)u(kh - h)\,, \tag{12.12}$$

where

$$\phi = \int_0^h e^{Ah}\,,$$

$$\Gamma_0(\tau_k) = \int_0^{h-\tau_k} e^{As} ds B \,,$$

and

$$\Gamma_1(\tau_k) = \int_{h-\tau_k}^{h} e^{As} ds B \,.$$

Proof. We apply equation (12.4) which yields:

$$\begin{aligned}
x(kh+h) =& e^{Ah}x(kh) + \int_{kh}^{kh+h} e^{A(kh+h-s)} Bu(s-\tau_k) ds \\
=& \phi x(kh) + \int_{kh}^{kh+\tau_k} e^{A(kh+h-s)} ds B u(kh-h) \\
&+ \int_{kh+\tau_k}^{kh+h} e^{A(kh+h-s)} ds B u(kh) \\
=& \{s' = kh+h-s\} \\
=& \phi x(kh) - \int_{h}^{h-\tau_k} e^{As'} ds' B u(kh-h) - \int_{h-\tau_k}^{0} e^{As'} ds' B u(kh) \\
=& \phi x(kh) + \Gamma_0(\tau_k) u(kh) + \Gamma_1(\tau_k) u(kh-h) \,.
\end{aligned}$$

□

Now we assume that the controller in the WSN-CS described above outputs a signal that is a linear function of the state, i.e. $u(x(kh)) = -Kx(kh)$ for some $K \in \mathbb{R}^{n\times n}$. In order to examine the stability of the resulting system, we would like to rewrite equation (12.12) in the form (12.6). To this end, we define the augmented state vector

$$z = \begin{bmatrix} x(kh) \\ u(kh-h) \end{bmatrix}$$

and the matrix

$$\overline{\phi}(\tau_k) = \begin{bmatrix} \phi - \Gamma_0(\tau_k)K & \Gamma_1(\tau_k) \\ -K & 0 \end{bmatrix} .$$

It can then be easily verified that equation (12.12) is equivalent to

$$z(kh+h) = \overline{\phi}(\tau_k) z(kh) \,. \tag{12.13}$$

Note that since $\overline{\phi}(\tau_k)$ in (12.13) depends on the time delay τ_k at time kh, the delayed system is no longer time-invariant. Thus the stability results that

we showed in section 12.1 are not applicable. However, if an appropriate network protocol is used, the delay τ_k will be constant, i.e. $\tau_k = \tau$ for all $k \in \mathbb{N}$. In this case, the system becomes time-invariant and we may apply theorem 12.1.5, from which we see that the stability of the WSN-CS is determined by the eigenvalues of $\overline{\phi}(\tau)$. We shall illustrate this with an example.

Example 12.3.2. *Suppose that we are given a simple scalar system that is subject to a constant network delay τ and governed by the following equation:*

$$\dot{x}(t) = u(t) \ .$$

Assume that the controller decision is $u(t) = -Kx(t)$.
In order to study the stability of this system, the matrix $\overline{\phi}$ needs to be constructed. Thus, since $A = 0$ and $B = 1$, we have that

$$\phi = e^0 = 1 \qquad \Gamma_0 = \int_0^{h-\tau} ds = h - \tau \qquad \Gamma_1 = \int_{h-\tau}^{h} ds = \tau$$

and the matrix becomes

$$\overline{\phi} = \begin{bmatrix} \phi - \Gamma_0(\tau) K & \Gamma_1(\tau) \\ -K & 0 \end{bmatrix} = \begin{bmatrix} 1 - hK + \tau & \tau \\ -K & 0 \end{bmatrix} .$$

We are now interested in computing the eigenvalues of $\overline{\phi}$:

$$\det\left(\overline{\phi} - \lambda I\right) = 0 \Leftrightarrow (1 - hK + \tau K - \lambda)(-\lambda) + \tau K = 0 \Leftrightarrow$$

$$\Leftrightarrow \lambda^2 - \lambda(1 - hK + \tau K) + \tau K = 0 \Leftrightarrow$$

$$\Leftrightarrow \lambda_{1,2} = \frac{1 - hK + \tau K \pm \sqrt{(1 - hL + \tau K)^2 - 4\tau K}}{2}$$

At this point we recall theorem 12.1.5 which tells us that the sampled system state is asymptotically stable if and only if $|\lambda_1|, |\lambda_2| < 1$.

If τ_k is greater than h for some k, the analysis becomes more complicated. This is because during a single sampling interval h, the plant may receive zero, one or more than one sampled signal. In the special case that there is some natural number $l > 1$ such that $(l-1)h < \tau_k < lh$ for all $k \in \mathbb{N}$, it can be shown by an argument similar to what was done in Example 12.3.2 that $w(kh + h) = \widetilde{\phi}(\tau_k) w(kh)$, where $\tau_k' = \tau_k - (lh - h)$ and

$$\widetilde{\phi}(\tau_k) = \begin{bmatrix} \phi & \Gamma_1(\tau_k') & \Gamma_0(\tau_k) & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & I \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \quad w(kh) = \begin{bmatrix} x(kh) \\ u(kh - lh) \\ \vdots \\ u(kh - 2h) \\ u(kh - h) \end{bmatrix}$$

Stability criterion for time-varying delay

When a contention-based MAC protocol is used, the system will be subject to a time-varying delay τ_k. Below we shall study the stability of a WSN-CS with time-varying delay, but for the moment we consider a non-delayed system with a discrete-time plant that is governed by the equations:

$$x(k+1) = Ax(k) + Bu(k) \tag{12.14}$$

$$y(k) = Cx(k) + Du(k)\,. \tag{12.15}$$

We assume that $x(0) = 0$ and thus a Z-transformation of the above equations yields:

$$zX(z) = AX(z) + BU(z) \tag{12.16}$$

$$Y(z) = CX(z) + DU(z) \tag{12.17}$$

where $X(z) = Z(x(k))$, $U(z) = Z(u(k))$ and $Y(z) = Z(y(k))$.

From (12.16) we have that $X = (zI - A)^{-1}BU$, which we insert into (12.17) and thus conclude that $Y = \left[C(zI - A)^{-1}B + D\right]U = HU$, where $H(z)$ is defined by $H = C(zI - A)^{-1}B + D$. The control law is assumed to be given as a convolution of the plant output $y(k)$ with some function $f(k)$. We make thus make use of the Z-transform again and conclude that $U(z) = F(z)Y(z)$

We are now ready to add network delay to the system we have considered. We shall assume that the system described above is subject to a time-varying delay τ_k such that $\tau_k \leq \tau_{\max}$, where $\tau_{max} \in \mathbb{N}$ is the maximum number of time-steps for which a packet can be delayed. For this situation it is possible to prove the following theorem:

Theorem 12.3.3. *Consider a WSN-CS subject to a time-varying delay $\tau_k \leq \tau_{\max}$ as described above. The system is stable if the following inequality holds for all $\omega \in [0, 2\pi]$:*

$$\left\| \frac{F(e^{i\omega})H(e^{i\omega})}{1 + F(e^{i\omega})H(e^{i\omega})} \right\| \leq \frac{1}{\tau_{\max}\left|e^{i\omega} - 1\right|}\,. \tag{12.18}$$

Proof. The proof is omitted. For a proof see (Kao and Lincoln, 2004). □

Note that $f(k)$ and $\tau_{\max}$ can be chosen such that condition (12.18) holds. This can be done by modifying the controller and the MAC protocol. Finally we remark that (12.18) is a sufficiency condition for stability, but it must not necessarily be satisfied by a stable system.

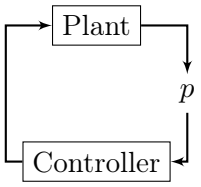


Figure 12.5 WSN-CS with packet loss and negligible delay

Choice of appropriate sampling interval of a system with delay

In a sampled system, in order to approximate the underlying continuous-time system as close as possible, is has conventionally been desired to have a short sampling period h. For a WSN-CS however, frequent sampling complicates the scheduling in the network and also leads to a higher energy consumption. Moreover, frequent sampling may increase the network load, which may lead to longer delays τ_k. As we can see from Example 12.3.2, the eigenvalues of $\bar{\phi}$(and hence the stability of the corresponding system), depends on both the delay τ and the sampling interval h. Thus the delay should be taken into consideration when deciding the sampling interval.

12.3.2 Packet losses

In this section, the effect of packet losses on the WSN-CS stability is examined. In order to simplify the analysis, we assume that the packet losses may only occur as the data packets are sent form the plant to the controller. We also assume that the network delay is negligible. As usual, the state equation for the system is assumed to be:

$$\dot{x}(t) = Ax(t) + Bu(t)\,.$$

The input $u(t)$ will depend on whether a packet drop has occurred or not and is given by: $u(t) = \overline{x}(kh)$ for $t \in [kh, kh + h)$, where

$$\overline{x}(kh) = \begin{cases} x(kh - h) & \text{the k:th packet was lost} \\ x(kh) & \text{otherwise}\,. \end{cases}$$

The probability of a packet loss is given by the **packet loss probability** and will be denoted by p. The basic set-up we have described is depicted in Figure **??**. Using Proposition 12.2.1 we arrive at the following system equation for the sampled system:

$$x(kh + h) = \phi x(kh) + \Gamma u(kh)\,.$$

Here ϕ and Γ are the usual matrices and $u(t)$ is as above. The following theorem gives criteria for stability of the sampled system.

Theorem 12.3.4. *Consider a WSN-CS with packet losses as described above. Assume that* $\rho(\phi - \Gamma K) < 1$.

(i) If $\rho(\phi) \leq 1$, *then the system is asymptotically stable for every* p.

(ii) If $\rho(\phi) > 1$, *then the system is asymptotically stable if*

$$\frac{1}{1 - \frac{\gamma_1}{\gamma_2}} < 1 - p\,,$$

where $\gamma_1 = \log(\lambda^2_{\max}(\phi - \Gamma K))$, $\gamma_2 = \log(\lambda^2_{\max}(\phi))$ *and* $\lambda_{\max}(A)$ *is the maximum eigenvalue of a matrix* A.

Proof. The proof is rather tedious and is not given here. The curious reader will find a proof in (Zhang et al., 2001). □

We remark that, the parameters p and L depend on factors that we may influence(we may for instance change MAC protocol or controller). Thus, given an unstable WSN-CS with packet losses, we may stabilize the system by modifying the parameters p and L such that the criteria in Theorem 12.3.4 hold.

12.3.3 Multiple-packet transmission

Due to bandwidth and packet size constraints, the plant output must sometimes be sent using multiple packets. In such cases, some of the packets may not reach the controller in time because of delay or packet losses. For simplicity we will only cover the case where the sampled plant state is transmitted in two packets, but it is not hard to extend the model to systems in which the state is sent using more than two packets. Again for simplicity, the network delay is assumed to be negligible.

First of all, we assume that the sampled state of the plant is given by

$$x\,(kh) = [x_1\,(kh)\, x_2\,(kh)]\ ,$$

where x_1 and x_2 are sent in different packets. Then, as depicted in Figure 12.6, we model the double-packet transmission with a switch that can be in two states. The position of the switch indicates which of the two packets that is sent at a particular time step. In our model, depending on the position of the switch, the auxiliary state vector $\overline{x}(kh)$ is updated according to the equations below:

$$\begin{array}{ll} S_1 : \overline{x}_1(kh) = x_1(kh) & \overline{x}_2(kh) = \overline{x}_2(kh-h) \\ S_2 : \overline{x}_1(kh) = x_1(kh-h) & \overline{x}_2(kh) = x_2(kh) \end{array}\ . \tag{12.19}$$

Assume now that the state equation for the sampled system looks almost like it usually does:

$$x(kh+h) = \phi x(kh) - \Gamma K \overline{x}(kh)\,. \tag{12.20}$$

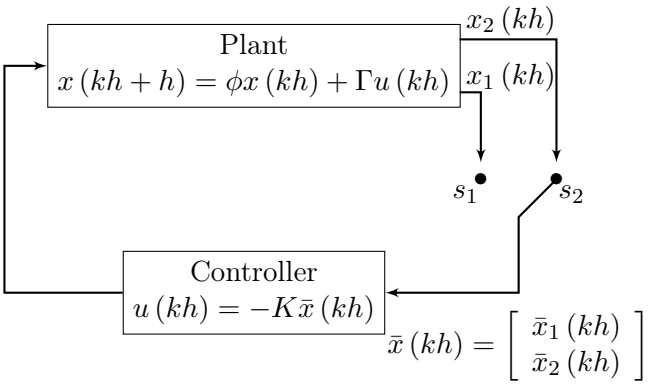


Figure 12.6 model of a WSN-CS with double packet transmission

Here

$$\overline{x}(kh) = \begin{bmatrix} \overline{x}_1(kh) \\ \overline{x}_2(kh) \end{bmatrix},$$

$$\phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix},$$

$$\Gamma = \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix},$$

and $K = [\, K_1 \; K_2 \,]$.

Defining an augmented state vector as

$$z(kh) = \begin{bmatrix} x_1(kh) \\ x_2(kh) \\ \bar{x}_1(kh) \\ \bar{x}_2(kh) \end{bmatrix},$$

it is an easy exercise to verify that the Equations in (12.19) and (12.20) are equivalent to:

$$z((k+1)h) = \tilde{\phi}_s z(kh) \quad \text{for} \quad s = 1, 2\,, \tag{12.21}$$

where $\tilde{\phi}_1$ and $\tilde{\phi}_2$ are given by

$$\tilde{\phi}_1 = \begin{bmatrix} \phi_{11} & \phi_{12} & -\Gamma_1 \mathrm{K}_1 & -\Gamma_1 \mathrm{K}_2 \\ \phi_{21} & \phi_{22} & -\Gamma_2 \mathrm{K}_1 & -\Gamma_2 \mathrm{K}_2 \\ \phi_{11} & \phi_{12} & -\Gamma_1 \mathrm{K}_1 & -\Gamma_1 \mathrm{K}_2 \\ 0 & 0 & 0 & I \end{bmatrix} \tag{12.22}$$

$$\tilde{\phi}_2 = \begin{bmatrix} \phi_{11} & \phi_{12} & -\Gamma_1 \mathrm{K}_1 & -\Gamma_1 \mathrm{K}_2 \\ \phi_{21} & \phi_{22} & -\Gamma_2 \mathrm{K}_1 & -\Gamma_2 \mathrm{K}_2 \\ 0 & 0 & I & 0 \\ \phi_{21} & \phi_{22} & -\Gamma_2 \mathrm{K}_1 & -\Gamma_2 \mathrm{K}_2 \end{bmatrix} \tag{12.23}$$

respectively.

Now let us turn back to the original issue that brought us here, i.e. that of stability. In order to simplify the analysis we assume that we have a scheduling network and that packets are sent in the following order: $x_1, x_2, x_1, x_2, \ldots$ In this case, we may describe the effect of sending the complete system state(i.e. two packages) by $x(kh + 2h) = \phi_2 x(kh + h) = \phi_2\phi_1 x(kh)$. Hence the study of the system stability is in this case reduced to the study of the eigenvalues of the the matrix product $\phi_2\phi_1$.

12.4 Sampling methods

One important way to reduce the energy consumption of a WSN-CS is to reduce the network communication. In particular we would like to reduce the idle listening time, since idle listening is a particularly energy consuming node activity. So far we have only considered systems with constant sampling periods. However, sampling methods that use aperiodic sampling could potentially reduce energy consumption. The following discussion is based on (Tiberi et al., 2013) and covers different kinds of such methods.

12.4.1 Event-triggered sampling

The first aperiodic sampling method we shall deal with is called **event-triggered sampling**. In this sampling method, samples are sent to the controller only when a certain kind of event occurs, i.e. when a specified function of the system state meets specific requirements. An example of such a requirement is that this function crosses a pre-determined threshold. With this sampling method, we can hope to decrease the average amount of communication through the network, since we take samples only when our requirements are met. However, this sampling method forces the nodes to perpetually wait for communication, i.e. the nodes must be in the highly energy consuming state of idle listening all the time. It is thus clear that we lose all energy savings that an aperiodic method potentially could have provided. Hence, by itself, event-triggered sampling is not a suitable sampling method for a WSN-CS.

12.4.2 Self-triggered sampling

The second sampling method to be examined is **self-triggered sampling**. With this method, an estimate of the evolution of the system state is computed at every sampling instant. In analogy with the method above, the next sampling instant is then set to be the point in time when a function of the system state meets specific requirements. Just like the previous method,

the sampling time is not periodic and the average amount of network communication could potentially be reduced. However, unlike the event-triggered sampling, the nodes of the network know when the next sample will be sent and can therefore avoid idle listening. Instead they may enter an energy saving mode while waiting for the next sample. At first sight, self-triggered sampling might seem ideal for WSN-CSs, but it has an important flaw. Indeed, every sampling instant is based merely on an estimation of the evolution of the state. Thus, unforeseen perturbations and uncertainties of the prediction may cause the system to behave unexpectedly and even make it unstable.

12.4.3 Adaptive self-triggered sampling

As we have learnt above, event-triggered and self-triggered sampling both fail to provide reliable and energy efficient sampling. However, a compromise between the two methods has been imagined: a sampling method that is capable of both predicting the evolution of the system and of detecting perturbations. Such a system would decrease the amount of network communication as well as the idle listening period length. In (Tiberi et al., 2013), one of the most recent works on this kind of sampling method is presented. In the model used there, an estimate of the system disturbance is considered in the computation of the next sampling instant. This leads to less conservative sampling intervals and to an improvement of the system response.

As we have explained, the the choice of sampling method is important, since it has influence on the energy consumption of the system. However, it is not enough to focus solely on the sampling method, instead the protocols of the network must also be designed to adapt to the sampling method. Reciprocally, the sampling method must take the restrictions caused by some protocols into account. For instance, IEEE 802.15.4, which is the most common MAC protocol used for WSNs, does not allow communication at every time t. This is an important constraint that must be addressed when designing the sampling method.

12.5 System design

WSN-CSs are complex systems that involve a lot of hardware and software with various functions. Designing such a system requires some organization. In this section, which is based on (Tiberi et al., 2013) and (Fischione et al., 2011), we shall discuss three different approaches for the design of a WSN-CS.

12.5.1 The Top-down approach

In the **top-down approach**, which is an approach traditionally used in the design of control applications, the structure of the network is not taken into account. The network is simply modelled as a black box that introduces perturbations such as delays, packet losses, etc. The control applications are designed to resist these perturbations, but they are assumed not to have any influence on them. The top-down approach presents several problems. Firstly, the models used to represent the network are often simplified and important constraints imposed by the network protocols are thereby neglected. Secondly, the energy-efficiency constraints imposed on the system by the presence of the WSN is not taken into account. Thirdly, the control applications are designed to entirely counter the imperfections created by the network, i.e. to withstand the worst case scenario(e.g. a high packet loss probability or long delays), even though this scenario might be unlikely. This leads to a waste of resources of the system and hence also to an energy-inefficient system.

12.5.2 The Bottom-up approach

The **bottom-up approach** on the other hand, is the approach traditionally used to design the protocols of a WSN. In this approach, the control applications are not explicitly considered, even though they are the center of the system. Instead, one simply tries to minimize the effect of usual problems inherent to a wireless system, like network delay and packet losses. As a result, the protocols reduce those problems, but they can be very energy-inefficient. This is because high reliability and low delays often demand a high energy consumption.

12.5.3 The System-level approach

The design approaches discussed above have a problem in common: they separate control applications and network protocols. By proceeding like this, not only do we ignore the energy constraints imposed by the network, but we also forget the fact that it is not necessary to completely suppress the irregularities introduced by the network. Indeed, the control applications can withstand a certain amount of delay and packet losses without becoming unstable. Minimizing the delay and the packet losses is therefore unnecessary. However, most protocols for wireless systems are designed according to the approaches mentioned above and thus they are too one-sided. For instance, the protocol RMST offers reliability, but performs poorly in terms of delay or energy and the protocols Fetch and Dozer are not designed for the high network traffic load of control systems. These protocols are thus not suited for control applications.

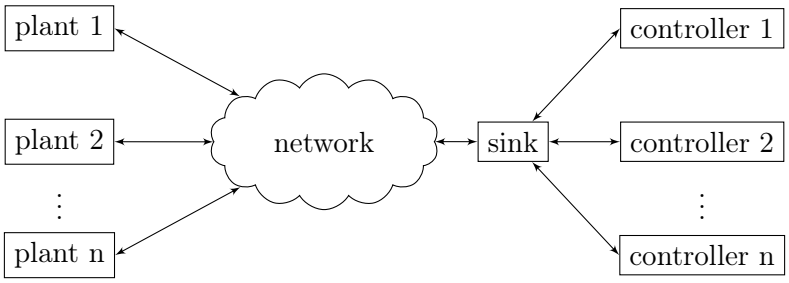


Figure 12.7 A WSN-CS with n plants and n controllers.

In order to tackle the problems with the top-down approach and the bottom-up approach, a new approach has been proposed to design WSN-CSs, namely the **system level approach** (Fischione et al., 2011). With this approach, one tries to find a trade-off between different parameters. These parameters include the network reliability and latency, as well as the energy efficiency. This is where the trade-off appears: a maximized latency and reliability leads to a high energy consumption, which is not possible with the low-power network we are using. When following the system level approach, one should strive for adaptability. Indeed the goal of the control applications may change overtime and it is necessary that the whole system is optimized to satisfy the new requirements. Furthermore, the network design must be easy to implement, since the nodes of the network have limited resources.

We shall now present a mathematical model of the trade-off introduced in the system-level approach. In order to do this we consider the problem of designing the protocols for a WSN that is used to close the loop between n plants and n controllers. State information from the sensors is sent to a common sink node, which is wired to the controllers. The set-up is illustrated in Fig. 12.7. For this situation, we may model the trade-off by the following optimization problem(Fischione et al., 2011):

$$\min_{x} E_{\text{tot}}(x) \tag{12.24}$$

$$\text{s.t.} \quad R_i(x) \geq \Omega_i\,, \quad i = 1, \ldots, n \tag{12.25}$$

$$\Pr[D_i(x) \leq \tau_i] \geq \Delta_i\,, \quad i = 1, ..., n\,. \tag{12.26}$$

Here x represents the variables of the system that can be modified, such as the radio power or the MAC parameters. E_{tot} represents the total energy consumed by the system. $R_i(x)$ is the probability for a packet sent from node i to reach the sink successfully. $D_i(x)$ is a random variable representing the delay of a packet sent from node i to the sink.

In the optimization problem above, the objective function to be minimized is the energy consumption, which makes is it a central aspect of the

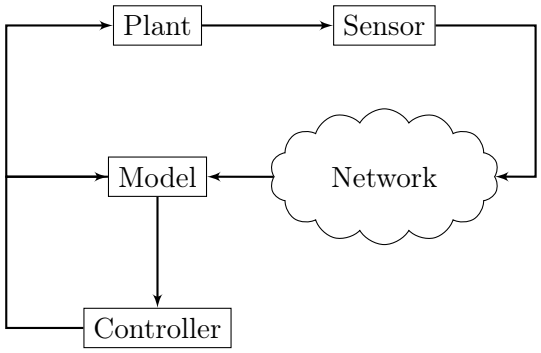


Figure 12.8 Set-up of a Model-Based NCS.

problem. This is what we are looking for. However, the system still has some requirements to meet when it comes to reliability and delay, Ω_i for the reliability, τ_i and Δ_i for the delay. This is where the trade-off between the network performances and the energy consumption is represented. We can also see that this mathematical problem allows adaptability. Indeed, if the requirements of the control applications change, then the new optimal x can be found by solving the optimization problem with the new set of constraints.

Unfortunately, the optimization problem above can be extremely difficult to solve. The expressions that link the different parameters can be non-linear and the solution may be difficult to find even for small networks. Hence, simplifications need to be made for this design to be possible. Such simplifications are implemented in the protocols Breath and TREnD, which are discussed further in (Fischione et al., 2011).

12.6 Model based network control system

Now we shall turn our attention to a type of a WSN-CS that we will refer to as a **model-based network control system (MB-NCS)**. Our analysis will follow (Montestruque and Antsaklis, 2004; Xie et al., 2014). The basic idea of the MB-NCS is to use a model of the plant that approximates the plant behaviour between sampling instants and thereby allow for a reduction of the packet transmission rate.

The set-up of the MB-NCS is depicted in Figure 12.8. We have a feedback control system consisting of a continuous-time plant, a discrete-time controller and a model of the plant, which is incorporated in the controller. At every sampling instant, the state of the model is set to be the actual plant state. During the rest of the time the control signal(and thus also the plant input signal) is determined by the plant model. There is also a wireless network present between the sensor and the controller which will be

responsible for packet losses and delay.

12.6.1 A model of the MB-NCS

Let us now make the above description of the MB-NCS precise by presenting the mathematical model used in (Xie et al., 2014). As before we assume that the system state of the plant evolves according to

$$\dot{x}(t) = Ax(t) + Bu(t)\,, \tag{12.27}$$

where $x(t)$ and $u(t)$ denotes the plant state and input respectively. Analogusly, the model of the plant is described by

$$\breve{x}(t) = \breve{A}\breve{x}(t) + \breve{B}u(t)\,, \tag{12.28}$$

where $\breve{x}(t)$ is the state of the model. $\breve{A}$ and $\breve{B}$ are estimates of A and B respectively. Furthermore, we assume that the model continuously updates the control signal to the plant according to $u(t) = -K\breve{x}(t)$. When a new sensor sample reaches the actuator, this sample is used as the state of the model:

$$\breve{x}\left(t_k\right) = x\left(t_k\right)\,. \tag{12.29}$$

In this model, we will not assume that the sampling period $h(k)$ is constant, but in order to simplify the analysis, we will still make the assumption that $t_0 = 0$. We now define the modelling error by $\bar{A} = A - \breve{A}$ and $\bar{B} = B - \breve{B}$. Similarly, the state error between the plant and the model is given by $e = x - \breve{x}$. We also define an augmented state vector $z(t)$ and a matrix Λ as follows:

$$z = \begin{bmatrix} x(t) \\ e(t) \end{bmatrix}, \quad \Lambda = \begin{bmatrix} A - BK & BK \\ \bar{A} - \bar{B}K & \breve{A} + \bar{B}K \end{bmatrix}. \tag{12.30}$$

With this new notation, we may describe the complete system(including both the plant and the model) by

$$\dot{z} = \Lambda z, \quad t \in [t_k, t_{k+1})\,. \tag{12.31}$$

Notice that (12.31) holds only for $t \in [t_k, t_{k+1})$ and hence, in order to use it to find $z(t)$, we also need an equation that specifies $z(t)$ at the sampling instants. From (12.32) it follows that the following equation holds for every k:

$$z(t_k) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} z(t_k^-)\,. \tag{12.32}$$

Proposition 12.6.2 below will provide a handy formula for calculating the system state, but first we need the following simple lemma:

Lemma 12.6.1. *The solution $z(t)$ of* (12.31) *and* (12.32) *satisfies:*

$$z(t_k) = \left(\prod_{j=0}^{k-1} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} \right) z_0 \,. \tag{12.33}$$

Proof. We proceed by induction over k. For $k = 0$ there is nothing to prove. Assume that (12.33) holds for a certain k and consider $k+1$: From (12.32) we have

$$z(t_{k+1}) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} z(t_{k+1}^-) \,.$$

Now we use (12.4) and (12.31), which together yield:

$$z(t_{k+1}) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda(t_{k+1}-t_k)} z(t_k) \,.$$

By the induction hypothesis we thus have

$$\begin{aligned} z(t_{k+1}) &= \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda(t_{k+1}-t_k)} \left(\prod_{j=0}^{k-1} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} \right) z_0 \\ &= \left(\prod_{j=0}^{k} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} \right) z_0 \,. \end{aligned}$$

Thus (12.33) holds for $k+1$ and the proof is complete. □

Proposition 12.6.2. *For $t \in [t_k, t_{k+1})$, the solution of* (12.31) *and* (12.32) *satisfies*

$$z\,(t) = e^{\Lambda(t-t_k)} \prod_{j=0}^{k-1} M\,(j) z_0 \tag{12.34}$$

where

$$z_0 = \begin{bmatrix} x(0) \\ 0 \end{bmatrix} ,$$

is the initial state and

$$M(j) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} .$$

Proof. Note that $\left[\begin{smallmatrix} I & 0 \\ 0 & 0 \end{smallmatrix}\right] e^{\Lambda h(j)} = \left[\begin{smallmatrix} S_j & P_j \\ 0 & 0 \end{smallmatrix}\right]$, for some matrices S_j and P_j. It follows that

$$\prod_{j=0}^{k-1} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} = \begin{bmatrix} \prod_{j=0}^{k-1} S_j & \prod_{j=0}^{k-1} P_j \\ 0 & 0 \end{bmatrix}$$

Thus, in light of Lemma 12.6.1

$$\begin{aligned} z(t_k) &= \left(\prod_{j=0}^{k-1} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} \right) z_0 \\ &= \left(\prod_{j=0}^{k-1} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} e^{\Lambda h(j)} \right) \begin{bmatrix} x(0) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \prod_{j=0}^{k-1} S_j & \prod_{j=0}^{k-1} P_j \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ 0 \end{bmatrix} = \begin{bmatrix} \prod_{j=0}^{k-1} S_j & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ 0 \end{bmatrix} \\ &= \left(\prod_{j=1}^{k-1} M(j) \right) z_0 \,. \end{aligned}$$

Now for $t \in [t_k, t_{k+1})$, (12.4) and (12.31) imply

$$z\,(t) = e^{\Lambda(t-t_k)} \prod_{j=1}^{k-1} M\,(j) z_0 \,,$$

and we are done. □

12.6.2 MB-NCS stability

Now we shall investigate the stability of the MB-NCS we have described. Firstly, we assume that the sampling intervals $h(k)$ are independently and identically distributed (i.i.d.) random variables with the common probability distribution F. This implies that the neither $h(k)$ nor $M(k)$ depend on k and we put $h(k) = h,\ M(k) = M$.

Now that random variables have entered into the picture, we need to reformulate our notion of stability. We will not give a very general definition, instead we shall say that the solution $z = 0$ of the system described by the equations (12.31) and (12.32) is **globally mean square asymptotically stable** if, for every initial-value $z_0 = z(0)$, the corresponding solution $z(t)$ of (12.31) and (12.32) satisfies:

$$E\left[||z(t)||^2\right] \xrightarrow[t\to\infty]{} 0\,. \tag{12.35}$$

Proposition 12.6.3. *Consider the system described by* (12.31) *and* (12.32)*, with sampling intervals $h(k)$ that are independently and identically distributed (i.i.d.) random variables with probability distribution F. The solution $z = 0$ of this system is globally mean square asymptotically stable if both of the following two inequalities hold:*

$$E_M = E\left[\left(e^{||\Lambda||h}\right)^2\right] < \infty \tag{12.36}$$

$$\left\| E\left[M^T M\right] \right\| < 1\,. \tag{12.37}$$

Proof. The proof is omitted. For a proof see (Montestruque and Antsaklis, 2004). □

Now we shall include packet losses in our stability analysis. Thus we assume that our network is subject to a packet loss probability p and we let $r = 1 - p$ denote the probability of a successful transmission. We assume furthermore that if a packet is lost, the sensor will attempt to send the packet again at the next sampling instant. We also make the simplifying assumption that the nominal sampling period(i.e the sampling period corresponding to the case when a packet is not lost) is constant and given by h_{nom}. Hence, the probability that the real sampling interval is $h = nh_{nom}$ is equal to $r(1-r)^{n-1}$, where $(n-1)$ is the number of consecutive packet losses. Now we see that the sampling intervals $h(k)$ are i.i.d. random variables and thus we may use Proposition 12.6.3. In the following theorem, which presents criteria for stability of the MB-NCS with packet losses, we will assume that Λ is diagonalizable. Specifically, we assume that $\bar{\Lambda} = P\Lambda P^{-1}$, where $\bar{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Theorem 12.6.4. *Consider the system described by* (12.31) *and* (12.32). *Assume that packet losses give rise to sampling intervals as described above. The solution $z = 0$ of this system is globally mean square asymptotically stable if* (12.38) *and* (12.39) *hold and*

$$(1-r)e^{2||\Lambda||h_{\text{nom}}} < 1 \tag{12.38}$$

$$\left\| \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} P^T S P \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right\| < 1\,, \tag{12.39}$$

$$S = E\left[e^{\bar{\Lambda}^T h} \left(P^{-1}\right)^T \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} P^{-1} e^{\bar{\Lambda} h} \right]. \tag{12.40}$$

Proof. We shall prove that (12.36) and (12.37) in Proposition 12.6.3 holds. We begin with (12.36):

$$\begin{aligned} E_M = E\left[\left(e^{||\Lambda||h} \right)^2 \right] &= \sum_{n=1}^{\infty} r(1-r)^{n-1} e^{2||\Lambda||nh_{\text{nom}}} \\ &= \sum_{n=1}^{\infty} r(1-r)^{-1} \left[(1-r)e^{2||\Lambda||h_{\text{nom}}} \right]^n. \end{aligned}$$

Thus it follows from (12.38) that $E_M < \infty$ and thus (12.36) holds.
Now we turn to (12.37):

$$\begin{aligned}
&\left\|E\left[M^T M\right]\right\| \\
&\quad = \left\|E\left[\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] e^{\Lambda^T h}\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right]\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] e^{\Lambda h}\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right]\right]\right\| \\
&\quad = \left\|E\left[\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] e^{\Lambda^T h}\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] e^{\Lambda h}\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right]\right]\right\| \\
&\quad = \left\|E\left[\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] P^T e^{\bar{\Lambda}^T h}\left(P^{-1}\right)^T\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] P^{-1} e^{\bar{\Lambda} h} P\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right]\right]\right\| \\
&\quad = \left\|\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] P^T E\left[e^{\bar{\Lambda}^T h}\left(P^{-1}\right)^T\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] P^{-1} e^{\bar{\Lambda} h}\right] P\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right]\right\| \\
&\quad = \left\|\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right] P^T S P\left[\begin{array}{cc} I & 0 \\ 0 & 0 \end{array}\right]\right\| < 1\,.
\end{aligned}$$

In the last inequality we have used (12.39).

□

Apart from packet losses, network delay is also important to consider when analysing the system stability. We may reduce the destabilizing effects of the network delay by time-stamping the packages. In this way, when the controller receives a delayed system sample it may estimate the current plant state as follows:

$$\breve{x}(t_k + \tau_k) = \left[\begin{array}{cc} I & 0 \end{array}\right] e^{\Lambda \tau_k}\left[\begin{array}{c} x(t_k) \\ e(t_k) \end{array}\right],$$

where τ_k is the delay of the k:th sample.

12.7 WSN-CS with Multiple Sensors

This last section is based on (Pajic et al., 2011) and is devoted to feedback control systems in which a wireless network is used in a way that is fundamentally different from how it has been used in the systems we have studied so far. We consider the case when the wireless control network has each network node acting as part of a controller in a feedback control system. In the specific set-up that we shall consider, a plant with multiple sensors and actuators is controlled by a multi-hop wireless network (see Figure 12.9). Note the difference between the WCN and the systems we have dealt with earlier in the chapter; in the WSN-CSs that we have studied in the sections above, information was routed to and from a single controller.

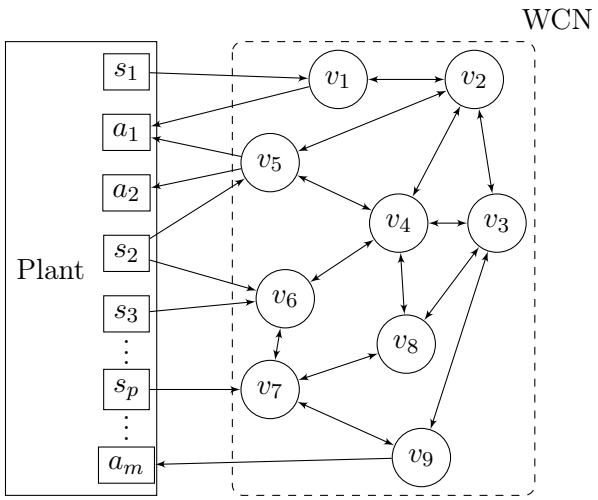


Figure 12.9 Set-up of the WSN-CS with distributed controllers over the network nodes (here, $N = 9$ nodes).

12.7.1 WCN Model

We shall now present the mathematical model of the WCN as described in (Pajic et al., 2011), but before we proceed to the actual model, we need some useful notation. A directed graph is an ordered pair $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is a set of vertices and $\mathscr{E}$ is a set of ordered pairs of different vertices, called directed edges. For a certain vertex $v \in \mathscr{V}$, the vertices in the set $\mathscr{N}_v = \{v' \in \mathscr{V} | (v', v) \in \mathscr{E}\}$ are said to be neighbours of the vertex v.

In order to describe the WCN, we construct a graph $\mathscr{G}$ as follows: Let the vertices be given by $\mathscr{V} = \mathscr{V}' \cup \mathscr{A} \cup \mathscr{S}$. Here $\mathscr{V}' = \{v_0, v_1, \ldots, v_N\}$ is the set of network nodes, $\mathscr{A} = \{a_0, a_1, \ldots, v_m\}$ is the set of actuators and $\mathscr{S} = \{s_0, s_1, \ldots, s_p\}$ is the set of sensors. The set of directed edges $\mathscr{E}$ is chosen so as to represent the radio connectivity, i.e. for any $\alpha, \beta \in \mathscr{V}$, we have $(\alpha, \beta) \in \mathscr{E}$ if and only if β can receive information directly from α.

The plant in the WCN is assumed to operate in discrete time and to be governed by the equations below:

$$\begin{aligned} x(kh+h) &= Ax(kh) + Bu(kh) \\ y(kh) &= Cx(kh)\,. \end{aligned} \tag{12.41}$$

As before, $x(kh) \in \mathbb{R}^n$ is the system state, $u(kh) \in \mathbb{R}^m$ is the plant input and $y(kh) \in \mathbb{R}^p$ is the plant output. In addition to the plant state vector $x(kh)$, we now also assume that every network node $v_i \in \mathscr{V}'$ have an associated scalar state $z_i(kh)$ that is updated at every time step according to the

following equation:

$$z_i(kh+h) = w_{ii}z_i(kh) + \sum_{v_j \in \mathcal{N}_{v_i}} w_{ij}z_j(kh) + \sum_{s_j \in \mathcal{N}_{v_i}} h_{ij}y_j(kh)\,. \qquad (12.42)$$

Hence, at every time-step the state of each node is set to be a linear combination of the network nodes and sensors from which it can receive information. In a similar fashion, we let the plant input $u_i(kh)$ be a linear combination of the node states that can send information to actuator a_i:

$$u_i(kh) = \sum_{j \in \mathcal{N}_{a_i}} g_{ij}z_j(kh)\,. \qquad (12.43)$$

The scalars w_{ij}, h_{ij} and g_{ij} are elements of the matrices $W \in \mathbb{R}^{N\times N}$, $\in \mathbb{R}^{N\times p}$ and $G \in \mathbb{R}^{m\times N}$ respectively. The radio connectivity imposes the following sparsity constraints on these matrices: $w_{ij} = 0$ if $v_j \notin \mathcal{N}_{v_i} \cup \{v_i\}$, $h_{ij} = 0$ if $s_j \notin \mathcal{N}_{v_i}$ and $g_{ij} = 0$ if $v_j \notin \mathcal{N}_{a_i}$. We let Φ denote the set of all three-tuples of matrices $(W, H, G) \in \mathbb{R}^{N\times N} \times \mathbb{R}^{N\times p} \times \mathbb{R}^{n\times N}$ that satisfies the sparsity constraints just described. If we now put together all node states into a value vector

$$z(kh) = \begin{bmatrix} z_1(kh) \\ z_2(kh) \\ \vdots \\ z_N(kh) \end{bmatrix}.$$

we may model the information exchange in the network, by

$$\begin{array}{rcl} z(kh+h) & = & Wz(kh) + Hy(kh) \\ u(kh) & = & Gz(kh) \end{array}\,. \qquad (12.44)$$

In order to incorporate the evolution of the plant state into the equation above, we put

$$\hat{x} = \begin{bmatrix} x(kh) \\ z(kh) \end{bmatrix}, \quad \hat{A} = \begin{bmatrix} A & BG \\ HC & W \end{bmatrix}\,.$$

With this notation, the system equation for the entire system becomes:

$$\hat{x}(kh+h) = \hat{A}\hat{x}(kh)\,. \qquad (12.45)$$

12.7.2 WSN-CS stability

We shall now address the question of how one should choose $(W, H, G) \in \Phi$ in order to make the system stable. In theory the answer is simple, we know that the system will be asymptotically stable if we choose $(W, H, G) \in \Phi$ so that

$$\rho(\hat{A}) < 1 \qquad (12.46)$$

Sometimes however, (12.46) is not satisfied for any $(W, H, G) \in \Phi$. If this is the case, then the plant cannot be stabilised with the given radio connectivity. If on the other hand it is possible to stabilize the plant, it is still not an easy problem to find suitable matrices W, H and G. The hardness of this problem is mainly due to the sparsity constraints imposed on these matrices. However, in (Pajic et al., 2011), the authors present an iterative algorithm that provides matrices such that (12.46) is satisfied. In the following theorem we describe this algorithm.

Theorem 12.7.1. *Consider the following algorithm:*

1. *Find feasible points X_0, Y_0, W_0, H_0, G_0 that satisfy the constraints:*

$$\begin{bmatrix} X_0 & \hat{A}^T \\ \hat{A} & Y_0 \end{bmatrix} \succ 0, \quad \begin{bmatrix} X_0 & I \\ I & Y_0 \end{bmatrix} \succeq 0, \quad \hat{A} = \begin{bmatrix} A & BG_0 \\ H_0C & W_0 \end{bmatrix}, \tag{12.47}$$

   where $(W_0, H_0, G_0) \in \Phi$ and $X_0, Y_0 \in \mathbb{R}^{(N+n)\times(N+n)}$ are positive symmetric matrices. If such feasible points do not exist, it is not possible to stabilize the plant with the given radio connectivity.

2. *At iteration $k(k \geq 0)$, from X_k, Y_k obtain the matrices $X_{k+1}, Y_{k+1}, W_{k+1}$, H_{k+1}, G_{k+1} by solving the following Linear Matrix Inequality(LMI) problem:*

$$\begin{aligned} \min \quad & \operatorname{tr}(Y_k X_{k+1} + X_k Y_{k+1}) \\ \text{s.t.} \quad & \\ & \begin{bmatrix} X_{k+1} & \hat{A}_{k+1}^T \\ \hat{A}_{k+1} & Y_{k+1} \end{bmatrix} \succ 0, \\ & \begin{bmatrix} X_{k+1} & \hat{A}^T \\ \hat{A} & Y_{k+1} \end{bmatrix} \succeq 0\,, \end{aligned}$$

   where

$$\hat{A} = \begin{bmatrix} A & BG_{k+1} \\ H_{k+1}C & W_{k+1} \end{bmatrix}$$

   and $(W_{k+1}, H_{k+1}, G_{k+1}) \in \Phi$ and $X_{k+1}, Y_{k+1} \in \mathbb{R}^{(N+n)\times(N+n)}$ are positive symmetric matrices.

3. *If $\rho(\hat{A}_{k+1}) < 1$, stop the algorithm. Otherwise set $k = k + 1$ and go to step 2.*

The algorithm described above determines a tuple $(W, H, G) \in \Phi$ such that $\rho(\hat{A}) < 1$ if the sequence $t_k = tr(Y_k X_{k+1} + X_k Y_{k+1})$ converges to $2(n + N)$.

Proof. The proof is omitted. For a proof see (Pajic et al., 2011). □

12.7.3 Advantages of the WCN

We conclude our review of the WCN with a discussion of the potential benefits of the WCN compared to a conventional system, i.e. a system with a single dedicated controller:

(i) **Computational inexpensiveness:** The WCN scheme presented above can be implemented on wireless nodes that are constrained in terms of computational power and memory storage. Indeed, every node in the network is required only to compute a linear combination of scalar node states.

(ii) **Simple scheduling:** In the WCN, the actuators do not need to wait for information packages to propagate all the way from the sensors. Instead all network nodes transmit their state information exactly once in each frame. This simplifies the scheduling of the communications in the network, since the only requirement on the communication is that it should be conflict-free.

(iii) **Short minimal sampling period:** Notice that the time-step k of the plant in (12.41) is the same as that of the network nodes in (12.42). Since the only requirement on the communication is that it should be conflict-free, the minimal sampling period is equal to $d_i T_{slot}$, where T_{slot} is the duration of a communication slot and d_i is the maximal degree of the interference graph $\mathscr{G}_{int} = (\mathscr{V}, \mathscr{E}_{int})$. By definition, there is an edge between to vertices in the interference graph iff their transmissions can interfere with each other.

(iv) **Multiple sensing/actuation points:** In real world control systems, there are often multiple sensors and actuators that are geographically distributed. The WCN, which do not rely on a single dedicated controller, is particularly well suited to handle these situations.

(v) **Compositionality:** Given a WCN with matrices (W_a, H_a, G_a) that controls a certain plant P_a, we may add another plant P_b and calculate a new set of stabilising matrices (W_b, H_b, G_b) for this plant. The important thing to note here is that calculation of (W_a, H_a, G_a) and (W_b, H_b, G_b) is completely decoupled. Each network node v_i may send its states $z_{a,i}$ and $z_{b,i}$ corresponding to P_a and P_b respectively, in a single packet. This of course generalises to the addition of more then one plant. Thus the WCN allows compositionality, in other words it is easy to extend the system with additional subsystems. This is in contrast to traditional network control systems, in which the addition of a new plant would require a complete recalculation of the entire network communication schedule.

Problems

Problem 12.1 Matrix Exponential

Let A be an $n \times n$ real or complex matrix. The exponential of A, denoted by e^A or $\exp(A)$, is the $n \times n$ matrix. Find e^A using two different way, where

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} .$$

Problem 12.2 Stability

Given a bi-dimensional state space system

$$X_{t+1} = \Phi X_t \,,$$

1. show how to compute the eigenvalues of Φ.
2. make some comments on the relationship between the eigenvalues of Φ and the stability.

Problem 12.3 Modeling

Model the dynamics of a coordinated turn (circle movement) using Cartesian and polar velocity. Here we assume that the turn rate ω is piecewise constant.

Problem 12.4 Linearized Discretization

In some cases, of which tracking with constant turn rate is one example, the state space model can be discretized exactly by solving sampling formula

$$x(t+T) = x(t) + \int_t^{t+T} a(x(\tau))d\tau \,,$$

analytically. The solution can be written as

$$x(t+T) = f(x(t)) \,.$$

Using this method, discretize the models in Ex:11.3.

Problem 12.5 Modeling of the Temperature Control

Assume that in winter, you'd like to keep the temperature in the room warm automatically by controlling a house heating system. Let T_i, T_o and T_r denote the temperature inside, outside and radiator. Thus the process model can be simplified as

$$\begin{aligned} \dot{T}_i =& \alpha_1(T_r - T_i) + \alpha_2(T_o - T_i) \\ \dot{T}_r =& \alpha_3(u - T_r) \,. \end{aligned}$$

1. Model the dynamics in standard state space form. Here assume that the outside temperature is aroud zero, $T_o = 0$.

2. Assume that the sampling time is h, model the continuous state space form to the discrete time standard form.

Problem 12.6 PID Controller

One heuristic tuning method for PID controller is formally known as the Ziegler-Nichols method. In this method, the K_i and K_d gains are first set to zero. The K_p gain is increased until it reaches the ultimate gain, G, at which the output of the loop starts to oscillate. G and the oscillation period T_G are used to set the gains, let $K_p = 0.60G$, $K_i = 2K_p/T_G$ and $K_d = K_pT_G/8$. Now consider the system in Ex: 11.3 and the step response plot shown in Fig. 12.10. Find T_G, then design the PID controller for the system in continuous space using Ziegler-Nichols method. Here assume that $G = 10$.

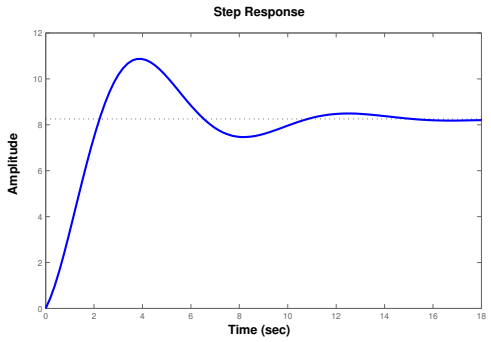


Figure 12.10 The step response for PID controller with $K_p = 12$.

Problem 12.7 Stability of Networked Control Systems with Network-induced Delay.

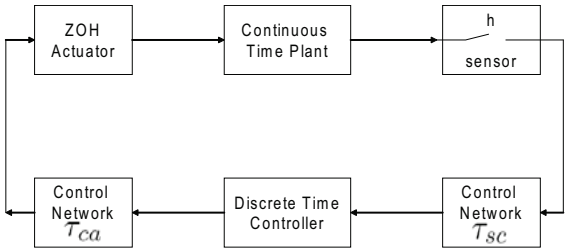


Figure 12.11 Networked Control System with communication delay.

Consider the Networked Control Systems (NCS) in Figure 12.11. The system consists of a continuous plant

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} ,$$

and a discrete controller

$$u(kh) = -Kx(kh), \qquad k = 0, 1, 2, \dots ,$$

where $A \in \mathbb{R}$, $B \in \mathbb{R}$, $C \in \mathbb{R}$.

Let $A = 0$, $B = I$. Illustrate the stability properties of the system as function of the network delays τ_{sc} and τ_{ca} under the assumptions that $\tau_{sc} + \tau_{ca} \leq h$ and that $h = 1/K$.

Problem 12.8 Control with time-varying delay

A process with transfer function

$$P(z) = \frac{z}{z - 0.5}$$

is controlled by the PI-controller

$$C(z) = K_p + K_i \frac{z}{z - 1} ,$$

where $K_p = 0.2$ and $K_i = 0.1$. The control is performed over a wireless sensor network, as shown in Figure 12.14. Due to retransmission of dropped packets, the network induces time-varying delays. How large can the maximum delay be, so that the closed loop system is stable?

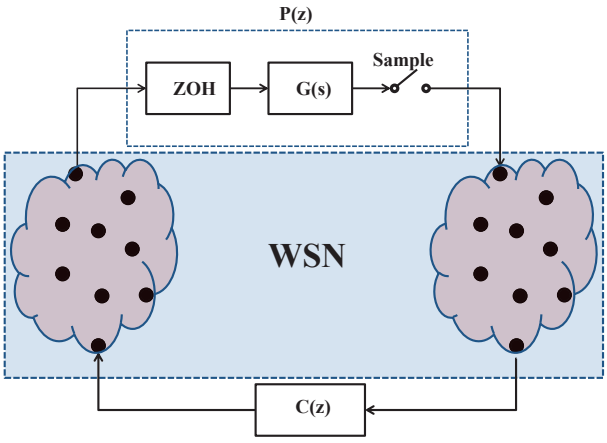


Figure 12.12 Closed loop system for Problem 12.2.

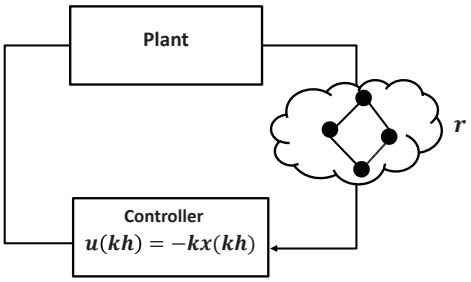


Figure 12.13 Networked Control System with packet losses.

Problem 12.9 Stability of Networked Control Systems with Packet Losses.

Consider the Networked Control System in Figure 12.13. It is assumed that the network is present only from the plant to the controller. The state space plant model is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

The feedback controller is $u(kh) = -Kx(kh)$, where $K = [20, 9]$.

Suppose that packets sent over the network are received at rate $r = 1-p$, where p is the packet loss rate, and that the system is sampled at rate $h = 0.3\,\text{s}$. What is the lower bound on reception rate r that still guarantee the stability of the system?

Problem 12.10 Networked Control System

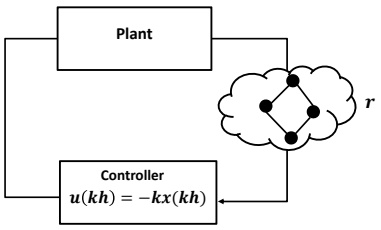


Figure 12.14 Closed loop system over a WSN.

Consider the Networked Control System (NCS) in Fig. 12.14. The system consists of a continuous plant

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{12.48a}$$
$$y(t) = Cx(t)\,, \tag{12.48b}$$

where $A = a$, $B = 1$, $C = 1$. The system is sampled with sampling time h, and the discrete controller is given by

$$u(kh) = -Kx(kh), \qquad k = 0, 1, 2, \dots\,,$$

where K is a constant.

(a) Suppose that the sensor network has a medium access control and routing protocols that introduce a delay $\tau \leq h$. Derive a sampled system corresponding to Eq.(12.48) with a zero-order-hold.

(b) Under the same assumption above that the sensor network introduces a delay $\tau \leq h$, give an augmented state-space description of the closed loop system so to account for such a delay.

(c) Under the same assumption above that the sensor network introduces a delay $\tau \leq h$, characterize the conditions for which the closed loop system becomes unstable [Hint: no need of computing numbers, equations will be enough]

(d) Now, suppose that the network does not induce any delay, but unfortunately introduces packet losses with probability p. Let $r = 1-p$ be the probability of successful packet reception. Give and discuss sufficient conditions for which the closed loop system is stable. If these conditions are not satisfied, discuss what can be done at the network level or at the controller level so to still ensure closed loop stability.

Problem 12.11 Model Based WSN-CS

Consider a MB-NCS where the state of the plant $x \in \mathbb{R}$ is governed by the equation $\dot{x}(t) = u(t)$. The input signal is given by $u(t) = -K\breve{x}(t)$, where $K \in \mathbb{R}$. We assume that the sampling interval is constant and put $h(k) = h$(and hence $t_k = kh$).

(a) How should we choose $\breve{A}$ and $\breve{B}$ in order to make this MB-NCS equivalent to a WSN-CS with zero-order hold(as described in section 12.2.2)?

(b) Let $\breve{A}$ and $\breve{B}$ be as in 12.11. Use Proposition 12.6.2 to compute the augmented system state $z(t)$ and verify that it agrees with Proposition 12.2.1.

Problem 12.12 Distributed WSN-CS

Consider a very simple WCN with only one sensor s_1, one actuator a_1 and one network node v_1 as in Figure 12.15. The scalar state and output of the plant is assumed to be governed by: $x(kh+h) = \alpha x(kh) + u(kh)$ and $y(kh) = x(kh)$, where $\alpha \in \mathbb{R}$.

(a) Construct the matrix $\hat{A}$ for the WCN described above.

(b) Show that for every α it is possible to choose h_{11}, g_{11} and w_{11} such that the system becomes asymptotically stable.

Problem 12.13 Energy-Efficient Control of NCS over IEEE 802.15.4 Networks (Tiberi et al., 2013).

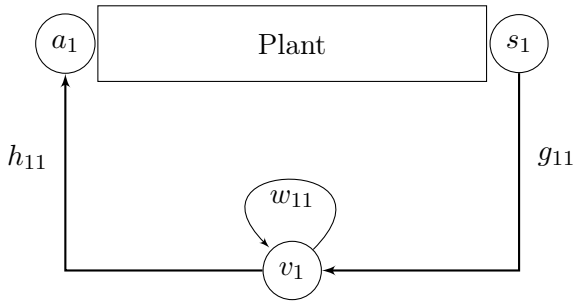


Figure 12.15 A simple WSN-CS.

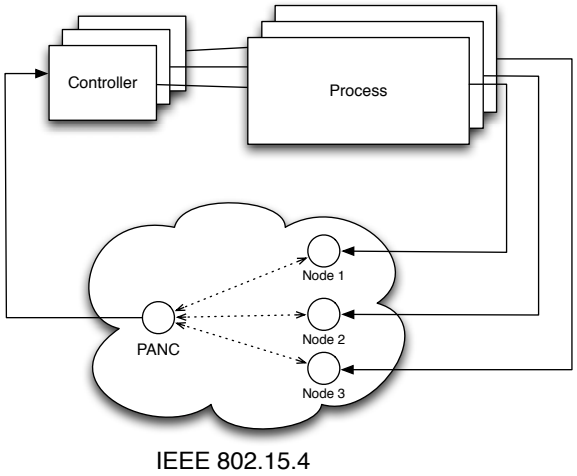


Figure 12.16 WAN-CS over IEEE 802.15.4 network.

Consider the Networked control system over IEEE 802.15.4 network composed of 3 control loops depicted in the Figure 12.16, where each process is scalar of the form $\dot{x}_i = a_i x_i + b_i u_i, i = 1, 2, 3$, and where the communication from the sensor nodes to the Personal Area Network Coordinator (PANC) is allowed only during the Guaranteed Time Slot (GTS) portion of the super-frame. Assume that there are no time delays, i.e. the transmissions from sensor i to the PANC and the respective control updates u_i are performed at the same instant $t = T_{i,k}$ and that each node can transmit only a packet per super-frame.

At each $t = T_{i,k}$, node i sends the values of $x_i(T_{i,k})$ and $t_{i,k+1}$ to the PANC, where $x_i(T_{i,k})$ is the measurement of the output of process i at time $t = T_{i,k}$, and $t_{i,k+1}$ is the time by which the next transmission from node i must be performed. The controller i updates the control input u_i with $u_i = -k_i x_i(T_{i,k})$ and it keeps it constant in the time interval $[T_{i,k}, T_{i,k+1})$. The transmissions are performed according to a self-triggered sampler that predicts the time in which the condition $|e_i(t)| := |x_i(T_{i,k}) - x_i(t)| \leq \delta_i$ is violated. The self-triggered sampler has the expression

$$t_{i,k+1} = T_{i,k} + \frac{1}{|a_i|} \ln \left(1 + \frac{|a_i| \delta_i}{|a_i - b_i k_i||x_i(T_{i,k})|} \right) . \tag{12.49}$$

Consider the numerical values of the described NCS as in the following table where $x_{i,0}$ denotes the initial condition of the process i. Determine:

	a_i	b_i	k_i	δ_i	$x_{i,0}$
Loop #1	2	1	?	$\frac{1}{2}$	5
Loop #2	3	-2	-2	?	8
Loop #3	2	$\frac{1}{2}$	6	$\frac{1}{2}$	?

(a) The values of k_1 such that practical-stability of the loop #1 is ensured.

(b) The values of δ_2 such that that practical-stability of the loop #2 is ensured.

(c) The values of $x_{3,0}$ such that that practical-stability of the loop #3 is ensured.

(d) For each control loop, find an upper-bound of the the practical-stability region size ε_i.

Appendix A

Random Variables

A.1 Basic Definitions

The basic concept in probability theory is that of a random experiment which is defined as an experiment whose outcome cannot be determined in advance, but is nevertheless still subject to analysis. Examples of random experiments are the toss of a dice, the numbering of the number of calls arriving at a telephone center during a fixed time period, etc. The sample space Ω of a random experiment is defined as the set of all possible outcomes of the experiment.

Often we are not interested in a single outcome but in whether or not one of a group of outcomes occurs. Such subsets of the sample space are called events and are usually denoted by capital letters $A, B, C, ...$ An event A occurs if the outcome of the experiment is one of the elements in A.

Since events are sets, we can apply the usual set operations to them:

1. The set $A \cup B$ (A union B) is the event that A or B occur.
2. The set $A \cap B$ (A intersection B) is the event that A and B both occur.
3. The event A^c is the event that A does not occur.
4. If $A \subset B$ (A is a subject of B) then event A is said to imply event B.

Two events A and B which have no outcomes in common, that is, $A \cap B = \emptyset$ are called disjoint events.

The probability of an event gives information about how likely it is that this particular event will occur. In specific, a probability P is a rule (function) which assigns a positive number to each event, and which satisfies the following axioms:

1. $\Pr(A) \geq 0$;
2. $\Pr(\Omega) = 1$;

3. For any sequence $A_1, A_2, ...$ of disjoint events we have

$$\Pr\left(\bigcup_i A_i\right) = \sum_i \Pr(A_i)$$

As a direct consequence of the axioms we have the following properties for P. Considering A and B as events, then

1. $\Pr(\emptyset) = 0$;
2. $\Pr \subset B \Rightarrow \Pr(A) \leq \Pr(B)$;
3. $\Pr(A) \leq 1$;
4. $\Pr(A^c) = 1 - \Pr(A)$;
5. $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$.

Probabilities may change when we know that an event has occurred. This leads to the definition of the conditional probability. Suppose an event $B \in \Omega$ has occurred. Since we know that the outcome lies in B, event A will occur if and only if $A \cap B$ occurs. Therefore, the conditional probability of A given B is

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)}.$$

We say A and B are independent if the knowledge that A has occurred does not change the probability that B occurs. That is

$$A, B \quad \text{independent} \Leftrightarrow \Pr(A|B) = \Pr(A)$$

Since

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)},$$

an alternative definition of independence is

$$A, B \quad \text{independent} \Leftrightarrow \Pr(A \cap B) = \Pr(A)\Pr(B).$$

Suppose $B_1, B_2, ...$ is a partition of Ω. That is, $B_1, B_2, ...$ are disjoint and their union is Ω. Then,

$$\Pr(A) = \sum_{i=1}^{n} \Pr(A \cap B_i),$$

and using the definition of the conditional probability we have

$$\Pr(A) = \sum_{i=1}^{n} \Pr(A|B_i)\Pr(B_i).$$

Using the above equation, we get the Bayes' rule

$$\Pr(B_j|A) = \frac{\Pr(A|B_j)\Pr(B_j)}{\sum_{i=1}^{n} \Pr(A|B_i)\Pr(B_i)}.$$

A.2 Random Variables

After completing the basic definitions regarding the random experiments, the concept of random variable is now introduced. Specifying a model for a random experiment via a complete description of Ω and P may not always be convenient or necessary. In practice we are only interested in various observations (i.e., numerical measurements) of the experiment. A random variable is a function from the sample space Ω to R.

Although random variables are, mathematically speaking, functions,it is often convenient to view random variables as observations of a random experiment that has not yet been carried out. Some examples of random variables without specifying the sample space are

1. The number of bugs in a computer program.
2. The total number of heads after tossing a coin n times.
3. The amount of time needed for an operation.

The set of all possible values a random variable X can take is called the range of X. We further distinguish between discrete and continuous random variables:

- Discrete random variables can only take isolated values. For example: a count can only take non-negative integer values.
- Continuous random variables can take values in an interval. For example: rainfall measurements, lifetimes of components, lengths,... are (at least in principle) continuous.

A.3 Probability Distribution

Let X be a random variable. We would like to specify the probabilities of events such as $\{X = x\}$ and $\{a \leq X \leq b\}$. If we can specify all probabilities involving X,we say that we have specified the probability distribution of X. One way to specify the probability distribution is to give the probabilities of all events of the form $\{X \leq x\},\ x \in R$. This leads to the following definition of the cumulative distribution function (cdf) of a random variable X as the function $F : R \to [0, 1]$

$$F(x) \stackrel{\Delta}{=} \Pr(X \leq x), \quad x \in \mathbb{R}.$$

The following properties for F are a direct consequence of the three axioms for P.

1. F is right-continuous:

$$\lim_{h \downarrow 0} F(x + h) = F(x).$$

2.
$$\lim_{x\to\infty} F(x) = 1$$
and
$$\lim_{x\to-\infty} F(x) = 0\,.$$

3. F is increasing: $x \le y \Rightarrow F(x) \le F(y)$.

4. $0 \le F(x) \le 1$.

Any function F with the above properties can be used to specify the distribution of a random variable X. Suppose that X has cdf F. Then the probability that X takes a value in the interval $(a, b]$ (excluding a, including b)is given by
$$\Pr(a < X \le b) = F(b) - F(a)\ .$$

The distribution of a random variable can be either discrete or continuous.

- We say that X has a discrete distribution if X is a discrete random variable. In particular, for some finite or countable set of values $x_1, x_2, ...$ we have
$$\Pr(X = x_i) > 0, \quad i = 1, 2$$
and
$$\sum_i \Pr(X = x_i) = 1\,.$$
The probability mass function (pmf) of X is then defined as
$$f(x) = \Pr(X = x)\,.$$

- A random variable X is said to have a continuous distribution if X is a continuous random variable for which there exists a positive function f with total integral 1, such that for all a, b
$$\Pr(a < X \le b) = F(b) - F(a) = \int_a^b f(u)\,du\,.$$
The function f is called the probability density function (pdf) of X. Note that the corresponding cdf F is simply a primitive of the pdf f. In particular,
$$F(X) = \Pr(X \le x) = \int_{-\infty}^{x} f(u)\,du\,.$$
Moreover, if a pdf f exists, then f is the derivative of the cdf F:
$$f(x) = \frac{d}{dx}F(x) = F'(x)\ .$$

Describing an experiment via a random variable and its pdf, pmf or cdf seems much easier than describing the experiment by giving the probability space.

Although all the probability information of a random variable is contained in its cdf (or pmf for discrete random variables and pdf for continuous random variables), it is often useful to consider various numerical characteristics of that random variable. One such number is the expectation of a random variable; it is a sort of "weighted average" of the values that X can take. A definition for both discrete and continuous cases follows:

- Let X be a discrete random variable with pmf f. The expectation (or expected value) of X, denoted by $\mathbb{E}\{X\}$, is defined by

$$\mathbb{E}\{X\} = \sum_x x \Pr(X = x) = \sum_x x f(x) .$$

- Let X be a continuous random variable with pdf f. The expectation (or expected value) of X, denoted by $\mathbb{E}\{X\}$, is defined by

$$\mathbb{E}\{X\} = \int_x x f(x)\, dx .$$

Another useful number about (the distribution of) X is the variance of X. This number, sometimes written as σ_X^2, measures the spread or dispersion of the distribution of X. The variance of a random variable X is defined by

$$\mathrm{Var}(X) = \mathbb{E}(X - \mathbb{E}\{X\})^2 .$$

The square root of the variance is called the standard deviation.

Some important discrete distributions are listed as following:

1. Bernoulli distribution: We say that X has a Bernoulli distribution with success probability p if X can only assume the values 0 and 1, with probabilities

$$\Pr(X = 1) = p = 1 - \Pr(X = 0) .$$

2. Binomial distribution: Consider a sequence of n coin tosses. If X is the random variable which counts the total number of heads and the probability of head is p then we say X has a binomial distribution with parameters n and p and write $X \sim Bin(n, p)$. The probability mass function X is given by

$$f(x) = \Pr(X = x) = \left(\begin{array}{c} n \\ x \end{array} \right) p^x (1-p)^{n-x}, \quad x = 0, 1, ..., n .$$

3. Geometric distribution: Again we look at a sequence of coin tosses but count a different thing. Let X be the number of tosses needed before the first head occurs. Then

$$\Pr(X = x) = (1-p)^{x-1}p, \quad x = 1, 2, 3... .$$

Such a random variable X is said to have a geometric distribution with parameter p and we write $X \sim G(p)$.

4. Poisson distribution: A random variable X for which

$$\Pr(X = x) = \frac{\lambda^x}{x!}e^{-\lambda}, \quad x = 0, 1, 2...$$

(for fixed $\lambda > 0$) is said to have a Poisson distribution and we write $X \sim \text{Poi}(\lambda)$.

Some important continuous distributions are the following:

1. Uniform distribution: We say that a random variable X has a uniform distribution on the interval $[a, b]$, if it has density function f,given by

$$f(x) = \frac{1}{b-a}, \quad a \leq x \leq b.$$

We write $X \sim U[a, b]$ where X can model a randomly chosen point from the interval $[a, b]$, where each choice is equally likely.

2. Exponential distribution: A random variable X with probability density function f,given by

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

is said to have an exponential distribution with parameter λ. We write $X \sim Exp(\lambda)$. The exponential distribution can be viewed as a continuous version of the geometric distribution.

3. Normal or Gaussian distribution: The normal (or Gaussian) distribution is considered as the most important distribution in the study of statistics. We say that a random variable has a normal distribution with parameters μ and σ^2 if its density function f is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad x \in R$$

We write $X \sim N\left(\mu, \sigma^2\right)$. The parameters μ and σ^2 turn out to be the expectation and variance of the distribution, respectively. If $\mu = 0$ and $\sigma = 1$ then

$$f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x}{2}^2}$$

and the distribution is known as a standard normal distribution.

4. Gamma and χ^2 distribution: The gamma distribution arises frequently in statistics. Its density function is given by

$$f(x) = \frac{\lambda^a x^{a-1} e^{-\lambda x}}{\Gamma(a)}, \quad x \geq 0$$

where Γ is the Gamma-function defined as

$$\Gamma(a) = \int_0^\infty u^{a-1} e^{-u} du, \quad a \geq 0\,.$$

Parameter a is called the shape parameter, and λ is called the scale parameter and we write $X \sim \text{Gam}(\alpha, \lambda)$. Of particular importance is following special case: A random variable X is said to have a chi-square distribution with n $(\in \{1, 2, ...\})$ degrees of freedom if $X \sim \text{Gam}(n/2, 1/2)$ and we write $X \sim \chi^2_n$

Appendix B

Sampling Theory

B.1 Sampling

One major issue in the application of WSNs for control tasks is the digitalization of the system. Usually our controlled systems are continuous in nature. For example the dynamics of a motor, the moisture and temperature drifts in a room or the concentration of ozone in the atmosphere are continuous systems.

WSNs introduce the problem of a sampled measurement. For a well behaved control of a system we generally require a high sampling rate. The energy efficiency paradigm in WSNs leads to the opposite requirement, since a low rate of measurements will cost less energy for the sensor and even more crucial for the transmission of the data. Therefore, it is worth to study the characteristics of sampled systems to develop methods to balance these two contradicting aims.

By sampling a continuous system we lose information. In the digitalized form we know the signal values only at the sampled time steps, but not at the points in between. We present here three standard procedures for converting continuous to discrete time. First of all the zero-order hold (ZOH) is a mathematical method for signal processing. It holds for each time interval a specified value. The mathematical formula is according to (Åström and Wittenmark, 1997)

$$f(t) = f(kh) \quad kh \leq t < kh + h\,.$$

A less used method is the first-order hold (FOH) procedure. It is slightly different to the ZOH method. A first-order hold filter transforms a continuous system in a piece by piece linear signal. The mathematical representation is given in (Åström and Wittenmark, 1997) as

$$f(t) = f(kh) + \frac{t - kh}{h}(f(kh + h) - f(kh)) \quad kh \leq t < kh + h.$$

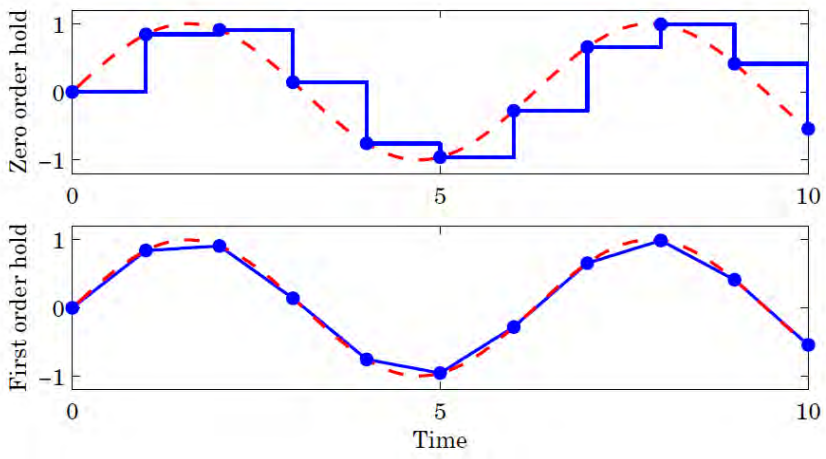


Figure B.1 Zero-order hold and First-order hold (Åström and Wittenmark, 1997). A sine signal (dashed, red) is sampled, indicated by the dots and the straight lined (blue) one is the reconstructed signal.

A good comparison between both transformations is given in Figure B.1.

The last method mentioned here is the so called Tustin transform. It maps the signal with

$$s = \frac{2}{T_s}\frac{z-1}{z+1},$$

from s-space to the z-space and explicitly assigns each value s in the analog plane to a corresponding discrete z value. Where T_s is the sampling rate which should be carefully selected as we will show soon. The Tustin transformation even allows to map a controller from the continuous design to a discrete approximation if the system is oversampled.

The question arises, which conclusion are possible with that reduced amount of information - or, which signals and frequencies can be reconstructed from the given sampled signal and which parameters do we need for that.

B.2 Reconstruction

As mentioned in the previous section, sampling is the process of converting analog signals in digital samples. Reconstruction is the reverse of this. Digital samples received from the sampling unit over an communication channel have to be converted back to a continuous analog signal. Therefore, based on the incoming samples, a certain voltage corresponding to the encoded value is generated at the output.

Nyquist-Shannon sampling theorem According to the Nyquist-Shannon sampling theorem a signal must be sampled at least 2 times during one period. That means, given a sampling rate ω_s, all signals with frequency $\omega \leq \omega_s/2$ can be constructed properly. This boundary is called Nyquist frequency and is defined as $\omega_N = \omega_s/2$. Signals greater than ω_N are transformed to lower frequencies and therefore distort the resulting output.

Aliasing and Anti-aliasing Filters The phenomenon when a high frequencies is interpreted as a low one is called aliasing. It appears if a signal is sampled that contains higher frequencies than Nyquist frequency. To minimize the effect of aliasing we can either use a high sampling frequency or add a lowpass filter before sampling. In short the aliasing appears if $\omega_s/2 > \omega_{\max}$ is not true, where $\omega_{\max}$ is the highest frequency component in the signal we want to sample.

B.3 Z-Transform

The Laplace transform is important in the input-output analysis of continuous time systems. The z-transform was introduced to play a similar role for sampled data systems and is a convenient tool to handle linear difference equations with or without initial values. Considering the discrete time signal $\{f(kh) \,:\, k = 0, 1, ...\}$, the z-transform of $f(kh)$ is defined as

$$Z\{f(kh)\} = F(z) = \sum_{k=0}^{\infty} f(kh)\, z^{-k}\,, \tag{B.1}$$

where z is a complex variable. The inverse transform is given by

$$f(kh) = \frac{1}{2\pi i} \oint F(z) z^{k-1} dz\,, \tag{B.2}$$

where the contour of integration encloses all singularities of $F(z)$. Some properties of the z-transform are summarized as follows.

1. Definition:
$$F(z) = \sum_{k=0}^{\infty} f(kh)\, z^{-k}\,;$$

2. Inversion:
$$f(kh) = \frac{1}{2\pi i} \oint F(z) z^{k-1} dz\,;$$

3. Linearity:
$$Z\{\alpha f + \beta g\} = \alpha Z\{f\} + \beta Z\{g\}\;;$$

4. Time shift:

$$Z\left\{q^{-n}f\right\} = z^{-n}F\,;$$

$$Z\left\{q^{n}f\right\} = z^{n}\left(F - F_1\right) \quad \text{where} \quad F_1\left(z\right) = \sum_{j=0}^{n-1} f\left(jh\right) z^{-j}\,;$$

5. Initial-value theorem:

$$f\left(0\right) = \lim_{z\to\infty} F\left(z\right)\,;$$

6. Final-value theorem:
   If $\left(1 - z^{-1}\right) F\left(z\right)$ does not have any poles on or outside the unit circle, then

$$\lim_{k\to\infty} f\left(kh\right) = \lim_{z\to 1}\left(1 - z^{-1}\right) F\left(z\right)\,;$$

7. Convolution:

$$Z\left\{f * g\right\} = Z\left\{\sum_{n=0}^{k} f\left(n\right) g\left(k-n\right)\right\} = Z\left\{f\right\} Z\left\{g\right\}\,.$$

Appendix C

Optimization Theory

C.1 Optimization Theory

Optimization problems arise in many different applications. They include the following elements:

- a mathematical model (discrete or continuous) that describes the problem of interest over some set of variables.
- a cost function of these variables that needs to be optimized according to some metric or norm.
- a set of constraints or conditions on the variables that need to be satisfied.

For example, the problem might be to determine the position of a source observed by several sensors. The model may include a stochastic description of the sources, noise, and propagation conditions. The optimization may be cast as a least squares problem, in which the expected variance of the position estimate is minimized. The constraints may include involvement of some maximum number of sensors or some maximum number of bits exchanged among the sensor nodes to conserve energy. In the following sections, a brief exposition of the basic tools of numerical analysis is presented, followed by a characterization of some classes of optimization problems and an outline of some classic approaches.

C.2 Basic Tools of Numerical Analysis

A basic fact of numerical methods is that linear problems are much easier to solve than non-linear ones. Consider, e.g., the problem of finding the roots (zeros) of the equation $f(x) = 0$. Now if the function were a line one could readily compute the point of intersection with the x-axis. Otherwise, the problem is typically approached by linearizing it and proceeding in a

sequence of iterations. For example, Newton's method begins by guessing the root as x_0, and forming (using Taylor's Theorem) the linear approximation at x_0:

$$l(x; x_0) = f(x_0) + f'(x_0)(x - x_0) ,$$

where $f'(x_0)$ denotes the first derivative. The unique root to the linear equation $l(x; x_0) = 0$ is, assuming the derivative is non-zero at x_0,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} .$$

One then proceeds through some series of iterations of the form

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})} ,$$

until some stopping condition is met, e.g., the difference between x_k and x_{k-1} is small enough or some maximum number of iterations is exceeded. Notice that Newton's method solves a non-linear problem through the following steps: approximation as a linear problem, solution of that linear problem, and iteration. This is a widely used general approach.

Consider now the problem of solving a system of non-linear equations:

$$\begin{aligned} &f_1(x_1, x_2, ..., x_n) = 0 \\ &f_2(x_1, x_2, ..., x_n) = 0 \\ &\vdots \\ &f_n(x_1, x_2, ..., x_n) = 0 . \end{aligned}$$

Using vector notation this may be rewritten as $\mathbf{f}(\mathbf{x}) = 0$ where $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ and $\mathbf{f} = (f_1, f_2, ..., f_n)^T$ Now in the particular case that the functions are linear and thus the system could be written as a matrix F times the vector $\mathbf{x}$ is equal to zero, then any of a number of efficient algorithms could be used to find the solution (e.g., QR decomposition, incomplete Gauss elimination) that have complexity of order n^3. Otherwise, the multidimensional form of Taylor's Theorem may be invoked to form the linear approximation about some vector $\mathbf{x}^{(k)}$:

$$\begin{aligned} &f_1(\mathbf{x}) \approx f_1\left(\mathbf{x}^{(k)}\right) + \nabla f_1\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x} - \mathbf{x}^{(k)}\right) \\ &f_2(\mathbf{x}) \approx f_2\left(\mathbf{x}^{(k)}\right) + \nabla f_2\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x} - \mathbf{x}^{(k)}\right) \\ &\vdots \\ &f_n(\mathbf{x}) \approx f_n\left(\mathbf{x}^{(k)}\right) + \nabla f_n\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x} - \mathbf{x}^{(k)}\right) \end{aligned} ,$$

where

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial x_n}\right) .$$

This may be written in matrix notation as

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}\left(\mathbf{x}^{(k)}\right) + \mathbf{J}_f\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x} - \mathbf{x}^{(k)}\right)$$

where the i-th row of the Jacobian matrix $\mathbf{J}_f(\mathbf{x})$ consists of $\nabla f_i(\mathbf{x})$. The kth iteration of Newton's method then consists of finding the solution $\mathbf{y}$ to the linear system of equations

$$\mathbf{J}_f\left(\mathbf{x}^{(k)}\right)\mathbf{y} = -\mathbf{f}(\mathbf{x})$$

where $\mathbf{y} = \mathbf{x}^{(k)} - x$ and $\mathbf{x}^{(k+1)} = \mathbf{y} + \mathbf{x}^{(k)}$.

C.3 Convex Optimizations

Optimization problems take the form

$$\begin{array}{ll} \min & f_0(\mathbf{x}) \\ \text{s. .t.} & f_i(\mathbf{x}) \leq b_i, \quad i = 1, \ldots, m\,, \end{array}$$

where the real-valued function f_0 is the objective function, the vector $\mathbf{x} = (x_1, x_2, ..., x_n)$ is the optimization variable, and the inequalities are the constraints of the problem. The optimal vector, x^*, produces the smallest value of the objective function such that all the constraints are also satisfied.

The optimization problem is an abstraction of the problem of making the best possible choice of a vector in R^n from a set of candidate choices. The variable $\mathbf{x}$ represents the choice made; the constraints $f_i(\mathbf{x}) \leq b_i$ represent firm requirements or specifications that limit the possible choices, and the objective value $f_0(\mathbf{x})$) represents the cost of choosing $\mathbf{x}$. A solution of the optimization problem corresponds to a choice that has minimum cost (or maximum utility), among all choices that meet the firm requirements.

Optimization problems are categorized in terms of the forms of the objective function and constraint functions. If f_0, f_1,. . . ,f_m satisfy

$$f_i(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha f_i(\mathbf{x}) + \beta f_i(\mathbf{y})\ ,$$

for all real numbers α, β and vectors **x,y**, then the problem is said to be a linear program. If this is not satisfied for any one of the functions the problem is said to be a non-linear program. A more significant distinction in practice is between convex and non-convex problems. A convex optimization problem is one in which the objective and constraint functions are convex, which means they satisfy the inequality

$$f_i(\alpha\mathbf{x} + \beta\mathbf{y}) \leq \alpha f_i(\mathbf{x}) + \beta f_i(\mathbf{y})$$

Linear programs are relatively easy to solve, and many convex problems are also fairly easily solved since there is but one global minimum with no

local minima that can trap iterative algorithms. By comparing the above equations, we see that convexity is more general than linearity: inequality replaces the more restrictive equality, and the inequality must hold only for certain values of α and β. Since any linear program is therefore a convex optimization problem, we can consider convex optimization to be a generalization of linear programming. Non-convex problems by contrast are usually very difficult. Thus in designing systems it is very advantageous to engineer the required optimizations to be convex or close enough that solutions can be reliably obtained. The least squares problem is the non-linear unconstrained optimization

$$\min \qquad f_0(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 ,$$

where $\mathbf{A}$ is an $k \times n$ matrix, $k > n$, $\mathbf{b}$ is a k-dimensional vector, and the squared Euclidean norm is used. The least squares problem fortunately is convex, and can be reduced to solving the set of linear equations

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} .$$

For least-squares problems we have good algorithms (and software implementations) for solving the problem to high accuracy, with very high reliability. The least-squares problem can be solved in a time approximately proportional to n^2k, with a known constant. A current desktop computer can solve a least-squares problem with hundreds of variables, and thousands of terms, in a few seconds; more powerful computers, of course, can solve larger problems, or the same size problems, faster. Algorithms and software for solving least-squares problems are reliable enough for embedded optimization. In many cases we can solve even larger least-squares problems, by exploiting some special structure in the coefficient matrix A. Suppose, for example, that the matrix A is sparse, which means that it has far fewer than kn nonzero entries. By exploiting sparsity, we can usually solve the least-squares problem much faster than order n^2k. A desktop computer can solve a sparse least-squares problem with tens of thousands of variables, and hundreds of thousands of terms, in around a minute (although this depends on the particular sparsity pattern). For extremely large problems (say, with millions of variables), or for problems with exacting real-time computing requirements, solving a least-squares problem can be a challenge. But in the vast majority of cases, we can say that existing methods are very effective, and extremely reliable.

C.4 Non-convex Optimizations

A large part of the difficulty in convex optimizations lies in casting the problem in the appropriate form. Once it has been realized that the problem is indeed convex, then there are several methods that can be used to find the solution systematically. For non-convex problems, it is by contrast relatively

easy to state the problem given there are so few standard forms for which efficient and reliable solution methods exist. It is then very hard actually to compute a solution reliably. However, convex optimization also plays an important role in problems that are not convex.

One obvious use is to combine convex optimization with a local optimization method. In local optimization, the compromise is to give up seeking the optimal x, which minimizes the objective over all feasible points. Instead we seek a point that is only locally optimal, which means that it minimizes the objective function among feasible points that are near it, but is not guaranteed to have a lower objective value than all other feasible points. Therefore, starting with a nonconvex problem, we first find an approximate, but convex, formulation of the problem. By solving this approximate problem, which can be done easily and without an initial guess, we obtain the exact solution to the approximate convex problem. This point is then used as the starting point for a local optimization method, applied to the original nonconvex problem.

Convex optimization is the basis for several heuristics for solving nonconvex problems. One interesting example we will see is the problem of finding a sparse vector $\mathbf{x}$ (i.e., one with few nonzero entries) that satisfies some constraints. While this is a difficult combinatorial problem, there are some simple heuristics, based on convex optimization, that often find fairly sparse solutions. Another broad example is given by randomized algorithms , in which an approximate solution to a nonconvex problem is found by drawing some number of candidates from a probability distribution, and taking the best one found as the approximate solution. Now suppose the family of distributions from which we will draw the candidates is parametrized, e.g. by its mean and covariance. We can then pose the question, which of these distributions gives us the smallest expected value of the objective? It turns out that this problem is sometimes a convex problem, and therefore efficiently solved.

Appendix D

Matrix Algebra

D.1 Matrix Inversion Formula

Proposition D.1.1. *For compatible matrices A, B, C and D,*

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\left(C^{-1}+DA^{-1}B\right)^{-1}DA^{-1},$$

assuming the inverses exist.

Proof. Begin by considering the block matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

By doing the LDU and UDL decomposition of M and equating them, we obtain

$$\begin{bmatrix} I & 0 \\ CA^{-1} & 0 \end{bmatrix}\begin{bmatrix} A & 0 \\ 0 & D-CA^{-1}B \end{bmatrix}\begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix} = \begin{bmatrix} I & BD^{-1} \\ 0 & I \end{bmatrix}\begin{bmatrix} A-BD^{-1}C & 0 \\ 0 & D \end{bmatrix}\begin{bmatrix} I & 0 \\ D^{-1}C & I \end{bmatrix}.$$

Thus inverting both sides yields

$$\begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix}\begin{bmatrix} A^{-1} & 0 \\ 0 & \left(D-CA^{-1}B\right)^{-1} \end{bmatrix}\begin{bmatrix} I & 0 \\ -CA^{-1} & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ -D^{-1}C & I \end{bmatrix}\begin{bmatrix} \left(A-BD^{-1}C\right)^{-1} & 0 \\ 0 & D^{-1} \end{bmatrix}\begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix}.$$

Equating the $(1,1)$ block shows

$$\left(A-BD^{-1}C\right)^{-1} = A^{-1} + A^{-1}B\left(D-CA^{-1}B\right)^{-1}CA^{-1}.$$

Finally substituting $C \to -D$ and $D \to C^{-1}$, we obtain

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\left(C^{-1}+DA^{-1}B\right)^{-1}DA^{-1}.$$

□

Appendix E

Graph Theory

E.1 Basic definitions

Definition E.1.1. A *graph* $G = (\mathscr{V}, \mathscr{E})$ is a pair of disjoint sets such that $\mathscr{E} \subseteq [\mathscr{V}]^2$, which is the set of all 2-elements subsets of $\mathscr{V}$. We call the elements of $\mathscr{V}$ *vertices* or *nodes* and the elements of $\mathscr{E}$ we call *edges*. We may picture a graph by drawing a dot for each vertex and if $\{u, v\} \in \mathscr{E}$ we draw this as a line between u and v.

Definition E.1.2. A *directed graph* $\overrightarrow{G} = (\mathscr{V}, \mathscr{A})$ is a pair of disjoint sets such that $\mathscr{A} \subseteq \mathscr{V} \times \mathscr{V}$. We call the elements of $\mathscr{A}$ *arcs* or *arrows* and picture $(u, v) \in \mathscr{A}$ by drawing an arrow from u to v.

Note that in Definition E.1.2 $(u, v) \neq (v, u)$.

Definition E.1.3. Let $G = (\mathscr{V}, \mathscr{E})$ be a graph (directed graph). If $G' = (\mathscr{V}', \mathscr{E}')$ is a graph (directed graph) such that $\mathscr{V}' \subseteq \mathscr{V}$ and $\mathscr{E}' \subseteq \mathscr{E}$, then we call G' a *subgraph* of G, and we write this as $G' \subseteq G$.

Definition E.1.4. A directed graph $G = (\mathscr{V}, \mathscr{A})$ is called *bidirectional* if for any $(u, v) \in \mathscr{A}$ we have that $(v, u) \in \mathscr{A}$.

Definition E.1.5. Given a graph $G = (\mathscr{V}, \mathscr{E})$, two vertices $u, v \in \mathscr{V}$ are said to be *neighbors* if $\{u, v\} \in \mathscr{E}$.

Definition E.1.6. The *degree* of a vertex $u \in \mathscr{V}$ is defined as

$$deg(u) = |\{v \in \mathscr{V} : \{u, v\} \in \mathscr{E}\}|$$

i.e. the degree of u is the number of neighbors of u.

Definition E.1.7. A *path* in a graph $G = (\mathscr{V}, \mathscr{E})$, is a non-empty graph P of the form

$$\mathscr{V}(P) = \{x_0, x_1, \ldots, x_n\},\ \mathscr{E}(P) = \big\{\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{n-1}, x_n\}\big\}.$$

We say that P is a path from x_0 to x_n and we refer to P as the sequence of $\mathscr{V}(P)$, that is $P = x_0 x_1 \ldots x_n$. By the *length* of P we mean $n - 1$.

Definition E.1.8. A *directed path* in a directed graph $\overrightarrow{G} = (\mathscr{V}, \mathscr{A})$, is a non-empty directed graph P of the form

$$\mathscr{V}(P) = \{x_0, x_1, \ldots, x_n\},\ \mathscr{A}(P) = \big\{(x_0, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n)\big\}.$$

As in Definition E.1.7 we refer to P as its sequence of vertices, that is $P = x_0 x_1 \ldots x_n$. By the *length* of P we mean $|\mathscr{A}(P)|$.

Definition E.1.9. A graph $G = (\mathscr{V}, \mathscr{E})$ is *connected* if, for any two $u, v \in \mathscr{V}$, there exists a path from u to v.

Definition E.1.10. A directed graph $\overrightarrow{G} = (\mathscr{V}, \mathscr{A})$ is *strongly connected* if, for any two $u, v \in \mathscr{V}$, there exists a directed path from u to v and a directed path from v to u.

Definition E.1.11. Let $G = (\mathscr{V}, \mathscr{E})$ be a graph. Two vertices u, v are said to be *k-connected* in G if and only if there are at least k distinct, node disjoint paths from u to v.

Definition E.1.12. A graph $G = (\mathscr{V}, \mathscr{E})$ is said to be *complete* if for any two vertices $u, v \in \mathscr{V}$, $\{u, v\} \in \mathscr{E}$.

Definition E.1.13. A *cycle* C is a path $P = x_0 x_1 \ldots x_n$ together with the edge $e = \{x_n, x_0\}$.

Definition E.1.14. A *forest* is a graph not containing any cycles.

Definition E.1.15. A *tree* is a connected forest.

Note that a tree T is a connected graph with $n - 1$ edges, where n is the number of vertices of T and if T is a plane graph then T has only one face, namely the outer face of T. Plane graphs and faces are defined below.

Definition E.1.16. Given a connected graph $G = (\mathscr{V}, \mathscr{E})$, a *spanning tree* of G is a tree $T = (\mathscr{V}, \mathscr{E}_T)$ such that $\mathscr{E}_T \subseteq \mathscr{E}$.

Definition E.1.17. Given a directed graph $\overrightarrow{G} = (\mathscr{V}, \mathscr{A})$, an *arborescence of $\overrightarrow{G}$ rooted at $u \in \mathscr{V}$*, is a directed graph $\overrightarrow{T} = (\mathscr{V}, \mathscr{A}_T)$ with $\mathscr{A}_T \subseteq \mathscr{A}$ and such that for any $v \in \mathscr{V}$ there exists a unique directed path from u to v.

Definition E.1.18. An *edge-weighted graph (arc-weighted directed graph)* is a graph (directed graph) $G = (\mathscr{V}, \mathscr{E})$ together with a function $w : \mathscr{E} \to \mathbb{R}^+$. We write $G = (\mathscr{V}, \mathscr{E}, w)$.

Definition E.1.19. Given an edge-weighted graph $G = (\mathscr{V}, \mathscr{E}, w)$, the *cost* C of G is defined as

$$C(G) = \sum_{e \in \mathscr{E}} w(\mathscr{E})\,.$$

Definition E.1.20. Given a connected edge-weighted graph $G = (\mathscr{V}, \mathscr{E}, w)$, a *minimal spanning tree* (MST) of G is a spanning tree $T = (\mathscr{V}, \mathscr{E}_T, w)$ such that for any other spanning tree T' of G we have that $C(T) \leq C(T')$.

Definition E.1.21. Let $G = (\mathscr{V}, \mathscr{E})$ be a connected graph where $\mathscr{V} \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$. A *Euclidean minimal spanning tree* (EMST) of G is a minimal spanning tree $T = (\mathscr{V}, \mathscr{E}_T, w)$ of G with edge-weight w defined by $w(\{u, v\}) = \delta(u, v)$, for any $\{u, v\} \in \mathscr{E}$.

Definition E.1.22. A *plane graph* is a pair $(\mathscr{V}, \mathscr{E})$ of sets with the following properties:

1. $\mathscr{V} \subset \mathbb{R}^2$,
2. every $e \in \mathscr{E}$ is an arc between two $v_1, v_2 \in \mathscr{V}$,
3. suppose $e_1 = \{u_1, u_2\} \in \mathscr{E}$ and $e_2 = \{v_1, v_2\} \in \mathscr{E}$. If $u_1 = v_1$ then $u_2 \neq v_2$ and if $u_2 = v_2$ then $u_1 \neq v_1$.
4. the interior of any $e \in \mathscr{E}$ contains no $v \in \mathscr{V}$ and no points of any $e' \in \mathscr{E}$ such that $e \neq e'$.

Definition E.1.23. Two graphs $G = (\mathscr{V}, \mathscr{E})$ and $H = (\mathscr{V}', \mathscr{E}')$ are said to be *isomorphic* if there is a bijection $\psi : \mathscr{V} \to \mathscr{V}'$ such that $\{u, v\} \in \mathscr{E}$ if and only if $\{\psi(u), \psi(v)\} \in \mathscr{E}'$.

Definition E.1.24. A graph is called *planar* if it is isomorphic to a plane graph.

Definition E.1.25. Let $G = (\mathscr{V}, \mathscr{E})$ be any plane graph. G divides $\mathbb{R}^2$ into a number of regions that we call *faces*. The unique face which contains a circle with all vertices and edges of G in its interior disc is called the *outer face* and all other faces are called *inner faces*.

Theorem E.1.26 (Menger 1927)**.** *Let $G = (\mathscr{V}, \mathscr{E})$ be a graph and $\mathscr{V}_1, \mathscr{V}_2 \subseteq \mathscr{V}$. Then the minimum number of vertices separating $\mathscr{V}_1$ from $\mathscr{V}_2$ is equal to the maximum number of disjoint $\mathscr{V}_1 - \mathscr{V}_2$ paths in G.*

For a proof of Theorem E.1.26, see (Diestel, 2005) page 62.

Definition E.1.27. Let $G = (\mathscr{V}, \mathscr{E})$ and let $U \subseteq \mathscr{V}$. We say that U is a *vertex cover* of $\mathscr{E}$ if every edge in G is incident with a vertex in U.

E.2 Proximity Graphs

Definition E.2.1. Given a graph $G = (\mathscr{V}, \mathscr{E})$, $\mathscr{V} \subset \mathbb{R}^2$ and two vertices $u, v \in \mathscr{V}$, $lune(u, v)$ is the intersection of the two discs of radius $\delta(u, v)$ centered at u and v respectively.

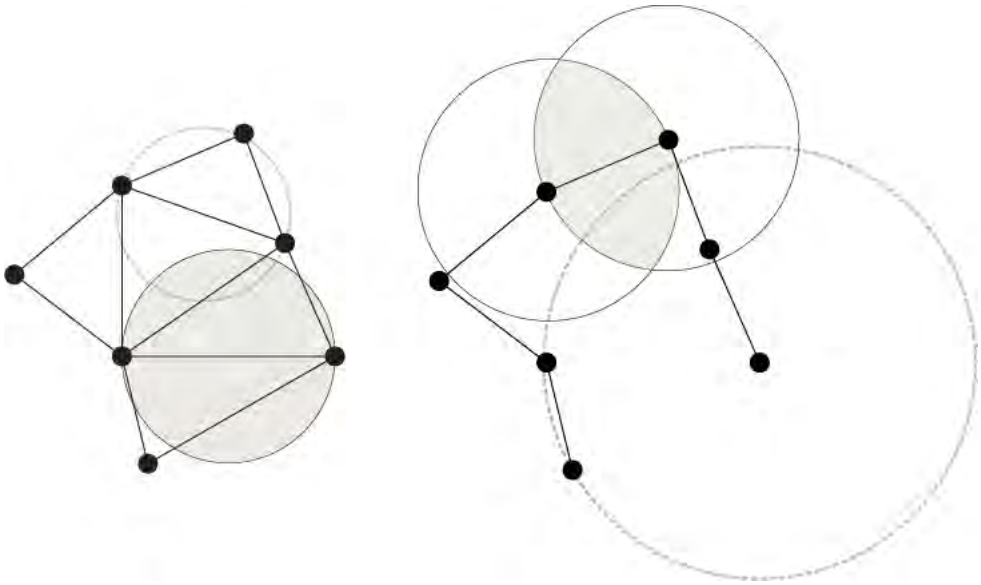


Figure E.1 Gabriel graph (left) and relative neighbor graph (right).

Definition E.2.2 (RNG)**.** Given a set $\mathscr{V} \subset \mathbb{R}^2$, the *Relative neighbor Graph* of $\mathscr{V}$, denoted by $RNG(\mathscr{V})$, is the graph $(\mathscr{V}, \mathscr{E})$ such that for any two vertices $u, v \in \mathscr{V}$, $\{u, v\} \in \mathscr{E}$ if and only if the interior of $lune(u, v)$ contains no vertex $v' \in \mathscr{V}$. Formally:

$$\forall u, v \in \mathscr{V}, \{u, v\} \in \mathscr{E} \text{ iff } \neg\exists w \in \mathscr{V} : \ \max\{\delta(u, w), \delta(v, w)\} < \delta(u, v)\,,$$

where $\neg$ means negation ("not").

Definition E.2.3 (GG)**.** Given a set $\mathscr{V} \subset \mathbb{R}^2$, the *Gabriel Graph* of $\mathscr{V}$, denoted by $GG(\mathscr{V})$, is the graph $(\mathscr{V}, \mathscr{E})$ such that for any two vertices $u, v \in \mathscr{V}$, $\{u, v\} \in \mathscr{E}$ if and only if the disc with diameter $\overline{uv}$ contains no vertex $v' \in \mathscr{V}$ in its interior. Formally:

$$\forall u, v \in \mathscr{V}, \{u, v\} \in \mathscr{E} \text{ iff } \neg\exists w \in \mathscr{V} : \delta(u, w)^2 + \delta(v, w)^2 < \delta(u, v)\,.$$

In Figure E.1 we see the GG and the RNG built on the same set of vertices.

Definition E.2.4 (DYG_k)**.** Given a set $\mathscr{V} \subset \mathbb{R}^2$ and a positive integer k, the *Directed Yao Graph* of $\mathscr{V}$, denoted by $DYG_k(\mathscr{V})$, is the graph $(\mathscr{V}, \mathscr{A}_k)$ constructed as follows. For each $u \in \mathscr{V}$, divide the plane into k sectors, each of angle $2\pi/k$, originated at u and denote the set of sectors by $\mathcal{S}_u = \{\mathscr{S}_{u,1}, \mathscr{S}_{u,2}, \ldots, \mathscr{S}_{u,k}\}$ ($\bigcup_i \mathscr{S}_{u,i} = \mathbb{R}^2$). For any $v \in \mathscr{V}$, $(u, v) \in \mathscr{A}_k$ if and only if $\exists \mathscr{S}_{u,i} \in \mathcal{S}_u$ such that for any vertex $w \in \mathscr{V}$ satisfying $w \in \mathscr{S}_{u,i}$, we have that $\delta(u, v) \leq \delta(u, w)$, i.e. v is the vertex, closest to u in $\mathscr{S}_{u,i}$.

Definition E.2.5 (YG_k)**.** Given a set $\mathscr{V} \subset \mathbb{R}^2$ and a positive integer k, the *Yao Graph* of $\mathscr{V}$, denoted by $YG_k(\mathscr{V})$, is the graph $(\mathscr{V}, \mathscr{E}_k)$ constructed as follows. Let $DYG_k(\mathscr{V}) = (\mathscr{V}, \mathscr{A}_k)$ be the Directed Yao Graph. For any two vertices $u, v \in \mathscr{V}$, $\{u, v\} \in \mathscr{E}_k$ if and only if $(u, v) \in \mathscr{A}_k$ and $(v, u) \in \mathscr{A}_k$.

Definition E.2.6. Let X be a metric space with metric d and $\mathscr{S}$ a set of points in X. Given $s \in \mathscr{S}$ the *Voronoi region* $\mathcal{V}(s)$ is defined as follows:

$$\mathcal{V}(s) = \{x \in X : d(x, s) \leq d(x, p), \forall p \in \mathscr{S}\}.$$

Definition E.2.7. Let X be a metric space with metric d and $\mathscr{S}$ a set of points in X. The *Voronoi diagram of* $\mathscr{S}$ is the graph $VD(\mathscr{S}) = (\mathscr{V}, \mathscr{E})$ such that for any two points $x, y \in X$, $x \in \mathscr{V}$ if and only if x is in the Voronoi region of at least 3 distinct points in $\mathscr{S}$ and $\{x, y\} \in \mathscr{E}$ if and only if the straight line between x and y is the intersection of two Voronoi regions of $\mathscr{S}$.

Definition E.2.8. Let $\mathscr{V}$ a set of points in $\mathbb{R}^2$. The *Delaunay graph* of $\mathscr{V}$, denoted by $DG(\mathscr{V})$, is the graph $G = (\mathscr{V}, \mathscr{E})$ such that for any two vertices $u, v \in \mathscr{V}$, $\{u, v\} \in \mathscr{E}$ if and only if $|\mathcal{V}(u) \cap \mathcal{V}(v)| > 1$.

Definition E.2.9 (DT)**.** Given a set $\mathscr{V} \subset \mathbb{R}^2$, a *Delaunay triangulation* of $\mathscr{V}$ is any triangulation $(\mathscr{V}, \mathscr{E})$ obtained by adding edges to the *Delaunay graph*.

Proposition E.2.10. *If $\mathscr{V}$ contains no four points on a circle, then the Delaunay graph $DG(\mathscr{V})$ is a triangulation. We denote this Delaunay triangulation by $DT(\mathscr{V})$.*

Proof. If $\mathscr{V}$ contains no four points on a circle, then there is no points contained in more than three Voronoi regions and hence every vertex in $VD(\mathscr{V})$ has degree three. Every face in $DG(\mathscr{V})$ corresponds to a vertex in $VD(\mathscr{V})$ and since every such vertex has degree 3, every face in $DG(\mathscr{V})$ must be a triangle. □

Theorem E.2.11. *Let $\mathscr{V}$ be a subset of $\mathbb{R}^2$ and $DG(\mathscr{V}) = (\mathscr{V}, \mathscr{E})$ the Delaunay graph of $\mathscr{V}$.*

1. *Three vertices $u, v, w \in \mathscr{V}$ are of the same face of $DG(\mathscr{V})$ if and only if the circle through u, v and w contains no points of $\mathscr{V}$ in its interior disc.*

2. *For any two vertices $u, v \in \mathscr{V}$, $\{u, v\} \in \mathscr{E}$ if and only if there is a closed disc which contains u and v on its boundary and contains no other vertex in $\mathscr{V}$.*

Proof. 1. If u, v, w are on the same face of $DG(\mathscr{V})$, then $\mathcal{V}(u) \cap \mathcal{V}(v) \cap \mathcal{V}(w) \neq \emptyset$. Let x be the unique point in $\mathcal{V}(u) \cap \mathcal{V}(v) \cap \mathcal{V}(w)$. Then $\delta(x, u) = \delta(x, v) = \delta(x, w) = d$ and there are no vertices closer than d to x. Hence the disc centered at x contains no vertices in its interior.

Conversely, assume that the circle C containing u, v, w contains no vertices in its interior disc. Let x be the center of C. Then $x \in \mathcal{V}(u) \cap \mathcal{V}(v) \cap \mathcal{V}(w)$ and hence u, v, w are on the same face of $DT(\mathscr{V})$.

2. Suppose that $\mathscr{D}$ is a closed disc containing u, v on its boundary but no other vertices. Let x be the center of $\mathscr{D}$. We want to show that $|\mathcal{V}(u) \cap \mathcal{V}(v)| > 1$. Since $x \in \mathcal{V}(u) \cap \mathcal{V}(v)$ we need only show that there is one more point in $\mathcal{V}(u) \cap \mathcal{V}(v)$. We know that x is either a vertex in $VD(\mathscr{V})$, or a point on an edge in $VD(\mathscr{V})$. By part *1*, the previous is not possible since $\mathscr{D}$ contains only 2 vertices on its boundary. Hence $|\mathcal{V}(u) \cap \mathcal{V}(v)| > 1$.

Conversely, assume that $\{u, v\} \in \mathscr{E}$. Then $|\mathcal{V}(u) \cap \mathcal{V}(v)| > 1$. Pick any point $p \in \mathcal{V}(u) \cap \mathcal{V}(v)$ that is not a vertex in $VD(\mathscr{V})$. Then the largest closed disc centered at p, that contains no vertices in its interior, contains only u and v on its boundary. □

Appendix F

WSNs Programming

F.1 TinyOS

Sensor networks are exciting emerging domain of deeply networked systems. The nodes of WSN are also known as wireless motes which are small low-powered devices having tiny amount of CPU and memory. Several kind of motes have been built with a drive towards miniaturization and low power. Eventual goal is to build micro-electromechanical systems(MEMS) sensors having dimension of 1 mm^3. The motes used in this course would be Berkeley Mica mote that are widely used by research groups all over the world.

Like other hardware systems (for example laptops, cell phones, etc.) OS and programming language are required to write any software for sensor nodes. TinyOS is a very small operating system specifically designed for sensor networks(especially popular in academia) whose core OS requires only 396 bytes of memory. It facilitates in building sensor networks application by providing important service and abstraction, such as sensing, communication, storage and timers. It has component oriented architecture and it defines a concurrent execution model, so developers can build applications out of reusable services and components without having to worry about unforseen interactions. It supports many components needing to act at the same time while requiring little RAM due to the concurrent execution model. Every I/O call in TinyOS is split phase: rather than block until completion, a request returns immediately and the caller gets a callback when the I/O completes. TinyOS only needs one stack, and not threads, as the stack is not tied up waiting for I/O calls to complete.

TinyOS and its applications are written in nesC(network embedded systems C) which is a dialect of C with support for components. The basic unit of nesC code is a **component**. Components connect via **interfaces**; they *use* and *provide* interfaces. So a component uses function of other components and provides its functionality to others as well. This is shown in Figure F.1

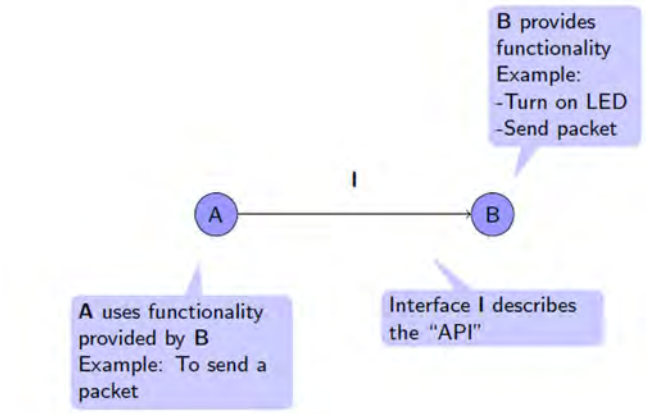
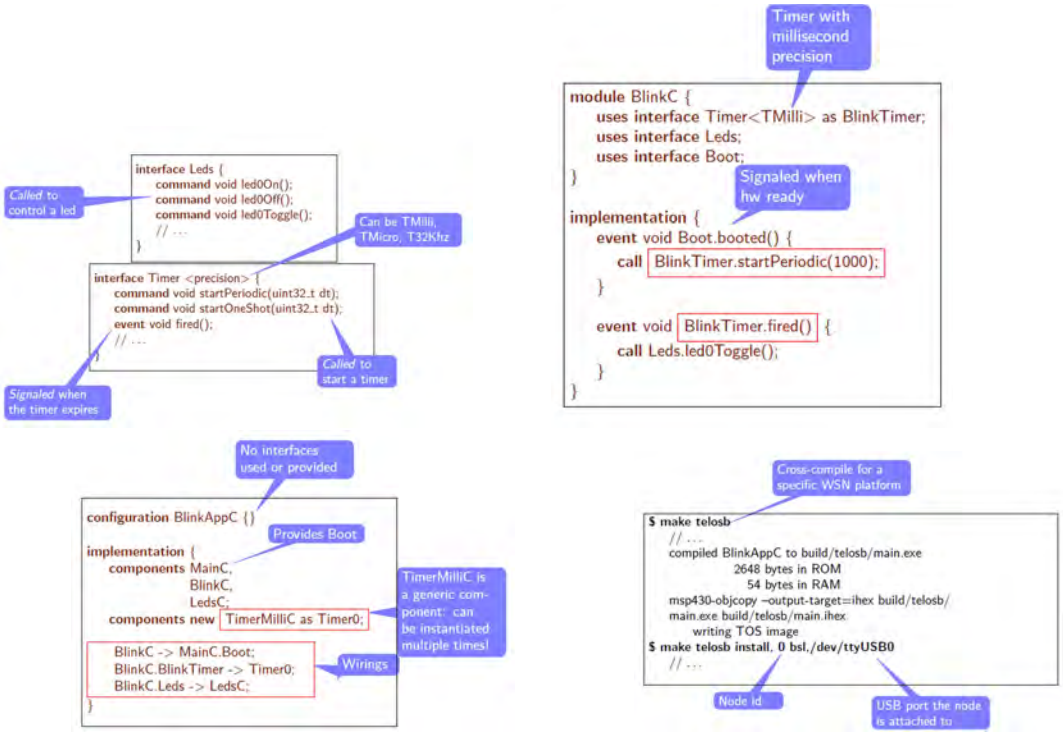


Figure F.1 nesC components and interfaces.

NesC interfaces are bidirectional and contain *commands* and *events*. A *command* is a function call from one component requesting service from another while an *event* is a function call indicating completion of service by a component. Inter-component protocols are elucidated by grouping of these commands/events together. Interfaces also contain the API for the current application.

There are two types of nesC components *modules* and *configurations*. *Modules* contain implementation code that is divided into 'Specification'and 'Implementation'. Specification lists the interfaces that are used/provided by the component and implementation contains *commands* of provided interfaces and *events* of used interfaces. *Configurations* are used to wire together components to create application. It can contain multiple sub-components, modules or configurations (nesting). Interface, module and configuration are exemplify next.

Example: Sample code for an application in nesC



Programming Tips

Here is a list of FAQs which might help you for programming tasks.

1. Download and install vmware player.
2. Download vmware image
3. Follow Installation howto(for both windows and linux).

How to run a test application

Now we want to connect a sensor node and run a test application

1. Connect the motes to the USB port and acticvate them. To connect to motes, you will have to tell the virtual machine that you would like it to recognize them. Once you have your motes connected and you have started XubunTOS, you can select which ones you would like to connect to. They should be listed at the top of the screen, with depressed buttons indicating a connection, and undepressed buttons indicating that no connection has been made. You may connect or disconnect them as you wish.
2. Go to test folder by typing following line in the Terminal environment
   $ cd /opt/tinyos-2.1.0/apps/Blink
3. Test if the motes are connected by following line
   $ motelist
4. Make and install the Blink app on the mote by following code
   $ make tmote install
   Now your mote should blink!!!

what is the Photo sensor class for telosb motes?

- HamamatsuS1087ParC

What Components do i need to be able to send and receive broadcast messages?

- components ActiveMessageC;
- components new AMSenderC(myID) as Sender;
- components new AMReceiverC(myID) as Receiver;

Which Interfaces do i need to implement for broadcasting?

- interface SplitControl as AMControl;
- interface Packet;
- interface AMSend;
- interface Receive;

How to wire message sending compenents into MyTheftApp?

- MyTheftAppC.AMControl ⇒ ActiveMessageC;
- MyTheftAppC.Packet ⇒ ActiveMessageC;
- MyTheftAppC.AMSend ⇒ Sender;
- MyTheftAppC.Receive ⇒ Receiver;

Reference

Use RadioSenseToLeds app under tinyOs totorial as a base for your implementations!.

Problems

Problem 6.1 Hello world

Implement a Hello world program in TinyOS. Implement a timer and toggle the blue LED every 2 sec.

Problem 6.2 Counter

Implement a counter using 3 LEDs. Use binary code to count-up every 1 seconds. Change the application to reset after it reaches 7.

Problem 6.3 Ping Pong

(a) Develop an application where two sensor nodes start to exchange a message in a ping pong manner. For this task you are not allowed to use Node IDs. (hint: probably you need to use broadcast message once. then upon receiving the message use unicast to ping pong message between sender and receiver.)

(b) Change the application such that only two nodes out of many nodes can ping pong. (hint: you might use a sequence number inside the packet!)

Problem 6.4 Dissemination Protocol

(a) The task is propagating a command in the sensor network. The command could be toggling a LED. Node ID 1 every 10 second sends a command to turn ON/OFF a selected LEDs. Receivers act accordingly and re-broadcast the command.

(b) How to avoid redundant commands? (hint: use a sequence counter to detect duplicate commands).

Bibliography

(2006). *Tmote Sky Data Sheet*. Moteiv, San Francisco, CA.

(2013). Principles of wireless sensor networks. [Online]. Available: https://www.kth.se/social/upload/510a5af2f2765403372ba230/lec3.pdf.

(2013). Rice distribution. [Online]. Available: http://en.wikipedia.org/wiki/Rice distribution.

(2020). About fading. [Online]. Available: http://wireless.agilent.com/wireless /helpfiles/n5106a/about fading.htm.

Akyildiz, I. F. and Vuran, M. C. (2010). *Wireless Sensor Networks*. Wiley.

Åström, K. J. and Wittenmark, B. (1997). *Computer Controlled Systems*. Prentice Hall, 3 edition.

Bahlmann, C., Haasdonk, B., and Burkhardt, H. (2002). Online handwriting recognition with support vector machines-a kernel approach. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 49–54. IEEE.

Bar-Shalom, Y. (1981). On the track-to-track correlation problem. *Automatic Control, IEEE Transactions on*, 26(2):571 – 572.

Bar-Shalom, Y. and Campo, L. (1986). The effect of the common process noise on the two-sensor fused-track covariance. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-22(6):803 –805.

Ben-Hur, A. and Noble, W. S. (2005). Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1):i38–i46.

Berg, T. and Durrant-Whyte, H. (1992). Distributed and decentralized estimation. In *Intelligent Control and Instrumentation, 1992. SICICI '92. Proceedings., Singapore International Conference on*, volume 2, pages 1118 –1123.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Cagalj, M., Hubaux, J., and Enz, C. (2002). Minimum-energy broadcast in all-wireless networks: Np- completeness and distribution issues. *Proc. ACM Mobicom 02, Atlanta, GA*, pages 172–182.

Carlson, N. (1990). Federated square root filter for decentralized parallel processors. *Aerospace and Electronic Systems, IEEE Transactions on*, 26(3):517 –525.

Chair, Z. and Varshney, P. (1986). Optimal data fusion in multiple sensor detection systems. *IEEE Transactions on Aerospace Electronic Systems*, 22(1).

Chang, K., Saha, R., and Bar-Shalom, Y. (1997). On optimal track-to-track fusion. *Aerospace and Electronic Systems, IEEE Transactions on*, 33(4):1271 –1276.

Chong, C. Y. (1979). Hierarchical estimation. In *Proceedings of the 2nd MIT/ONR C^3 Workshop*.

Chong, C.-Y., Mori, S., Barker, W., and Chang, K.-C. (2000). Architectures and algorithms for track association and fusion. *Aerospace and Electronic Systems Magazine, IEEE*, 15(1):5 –13.

Chong, C.-Y., Mori, S., and Chang, K.-C. (1985). Information fusion in distributed sensor networks. In *American Control Conference, 1985*, pages 830 –835.

Clemeniti, A., Crescenzi, P., Penna, P., Rossi, G., and Vocca, P. (2001). A worst-case analysis of an MST-based heuristic to construct energy-efficient broadcast trees in wireless sensor networks. *Tech. Report 010, Univ. of Rome "Tor Vergata", Math Department, Italy*, pages 172–182.

D. Willner, C. B. C. and Dunn, K. P. (1976). Kalman filter algorithms for a multisensor system. In *Proceedings of the 15th Conference on Decision and Control*.

Dargie, W. and Poellabauer, C. (2010). *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wiley.

de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2008). *Computational Geometry*. Springer, 3:rd edition.

Diestel, R. (2005). *Graph Theory*. Springer, 3:d edition.

Drummond, O. E. (1997). Tracklets and a hybrid fusion with process noise. In *Proceedings of the SPIE, Signal and Data Processing of Small Targets*.

Fischione, C., Park, P., Di Marco, P., and Johansson, K. H. (2011). *Wireless Network Based Control*, chapter Design Principles of Wireless Sensor Networks Protocols for Control Applications, pages 203–237. Springer.

Fuchs, B. (2006). On the Hardness of Range Assigmnment Problems. *Algorithms And Complexity*, 3998:127–138.

Garey, M. R. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1:st edition.

Garey, M. R. and Johnson, D. S. (1977). The Rectilinear Steiner Tree Problem is NP-Complete. *SIAM J. APPL. MATH.*, pages 826–834.

Gay, D., Levis, P., and Culler, D. (2005). Software Design Patterns for TinyOS. *ACM LCTES*.

Glad, T. and Ljung, L. (1981). *Reglerteknik : grundläggande teori*. Springer, 1 edition.

Gupta, V., Hassibi, B., and Murray, R. M. (2007). Optimal LQG control across packet-dropping links. *Systems and Control Letters*, 56(6):439–446.

Gupta, V. and Martins, N. C. (2009). On fusion of information from multiple sensors in the presence of analog erasure links. In *Proceedings of the IEEE Conference on Decision and Control (CDC), 2009*.

Gupta, V., Martins, N. C., and Baras, J. S. (2009). Stabilization over erasure channels using multiple sensors. *IEEE Transactions on Automatic Control*, 54(7):1463–1476.

Gustafsson, F. (2012). *Statistical Sensor Fusion*. Studentlitteratur, 2nd edition.

Hashemipour, H., Roy, S., and Laub, A. (1988). Decentralized structures for parallel kalman filtering. *Automatic Control, IEEE Transactions on*, 33(1):88 –94.

Hassan, M., Salut, G., Singh, M., and Titli, A. (1978). A decentralized computational algorithm for the global kalman filter. *Automatic Control, IEEE Transactions on*, 23(2):262 – 268.

Haykin, S. and Liu, K. R. (2009). *Handbook on array processing and sensor networks*. John Wiley & Sons.

Hovareshti, P., Gupta, V., and Baras, J. S. (2007). On sensor scheduling using smart sensors. In *Proceedings of the IEEE Conference on Decision and Control (CDC), 2007*.

Huang, C. and Tseng, Y. (2005). The Coverage Problem in a Wireless Sensor Network. *Mobile Netw. Appl.*, pages 519–528.

Jadbabaie, A., Lin, J., and Morse, A. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988 – 1001.

Kallik, R. (2010). A new statistical model of the complex nakagami-m fading gain. *IEEE Transactions on Communications*, 58(9).

Kao, C.-Y. and Lincoln, B. (2004). Simple stability criteria for systems withtime-varying delays. *Automatica*.

Kirousis, L. M., Kranakis, E., Krizanc, D., and Pelc, A. (2000). Power consumption in packet radio networks. *Theoretical Computer Science*, 243:289–305.

Kumar, S., Lao, T. H., and Arora, A. (2005). Barrier Coverage with Wireless Sensors. *Proc. AMC Conf. Mobile Comput. Netw.*, pages 284–298.

Levy, B. C., Castañon, D. A., Verghese, G. C., and Willsky, A. S. (1983). A scattering framework for decentralized estimation problems. *Automatica*, 19(4):373 – 384.

Li, M., Cheng, W., Liu, K., He, Y., Li, X., and Liao, X. (2012). Sweep Coverage with Mobile Sensors. *Proc. IEEE Int. Symp. Parallel Distrib. Process*, pages 1611–1619.

Li, M., Li, Z., and Vasilakos, A. V. (2013). A Survey on Topology Control in Wireless Sensor Networks: Taxonomy, Comparative Study, and Open Issues. *Proceedings of the IEEE*, 101(12):2538–2557.

Li, X., Wan, P., and Wang, Y. (2003). Integrated Coverage and Connectivity for Energy Conservation in Wireless Sensor Networks. *Proc. Tenth International Conf. on Comput. Comm. and Netw*, pages 564–567.

Liggins, M.E., I., Chong, C.-Y., Kadar, I., Alford, M., Vannicola, V., and Thomopoulos, S. (1997). Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85(1):95 – 107.

Luo, Z. Q. (2005a). An Isotropic Universal Decentralized Estimation Scheme for a Bandwidth Constrained Ad Hoc Sensor Network. *Selected Areas in Communications, IEEE Journal on*, 23(4):735–744.

Luo, Z. Q. (2005b). Universal Decentralized Estimation in a Bandwidth Constrained Sensor Network. *IEEE Transactions on Information Theory*, 51(6):2210–2219.

Mao, X., Miao, X., He, Y., Li, X., and Liu, Y. (2012). CitySee: Urban CO_2 Monitoring with Sensors. *Proc. IEEE INFOCOM*, pages 1611–1619.

Marina, M. and Das, S. (2002). Routing performance in the presence of unidirectional links in multihop wireless networks. *Proc. ACM Mobihoc 02*, pages 12–23.

Mo, L., He, Y., Liu, Y., Zhao, J., Tang, S., Li, X., and Dai, G. (2009). Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest. *Proc. AMC Conf. Embedded Netw. Sens. Syst.*, pages 99–112.

Montestruque, L. A. and Antsaklis, P. J. (2004). Stability of model-based networked control systems with time-varying transmission times. *IEEE Transactions on Automatic Control*.

Mori, S., Barker, W., Chong, C.-Y., and Chang, K.-C. (2002). Track association and track fusion with nondeterministic target dynamics. *Aerospace and Electronic Systems, IEEE Transactions on*, 38(2):659 –668.

Niu, R. and Varshney, P. (2005). Distributed detection and fusion in a large wireless sensor network of random size. *EURASIP Journal on Wireless Communications and Networking*, 4:462–472.

Olfati-Saber, R., Fax, J. A., and Murray, R. M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233.

Olfati-Saber, R. and Murray, R. (2004). Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520 – 1533.

Pajic, M., Sundaram, S., Pappas, G. J., and Mangharam, R. (2011). The wireless control network: A new approach for control over networks. *IEEE Transactions on Automatic Control*.

Pantos, G., Kanatas, A., and Constantinou, P. (2008). *Mobile Communication Systems*. Papasotiriou.

Pottie, G. and Kaiser, W. (2005). *Principles of Embedded Networked Systems Design*. Cambridge University Press.

Predd, J. B., Kulkarni, S. R., and Poor, H. V. (2005). Regression in sensor networks: Training distributively with alternating projections. In *Optics & Photonics 2005*, pages 591006–591006. International Society for Optics and Photonics.

Predd, J. B., Kulkarni, S. R., and Poor, H. V. (2006a). Consistency in models for distributed learning under communication constraints. *Information Theory, IEEE Transactions on*, 52(1):52–63.

Predd, J. B., Kulkarni, S. R., and Poor, H. V. (2006b). Distributed kernel regression: An algorithm for training collaboratively. In *Information Theory Workshop, 2006. ITW'06 Punta del Este. IEEE*, pages 332–336. IEEE.

Rao, B. and Durrant-Whyte, H. (1991). Fully decentralised algorithm for multisensor kalman filtering. *Control Theory and Applications, IEE Proceedings D*, 138(5):413 –420.

Ren, W. and Beard, R. (2005). Consensus seeking in multiagent systems under dynamically changing interaction topologies. *Automatic Control, IEEE Transactions on*, 50(5):655 – 661.

Ribeiro, A., Giannakis, G. B., and Roumeliotis, S. (2006). SOI-KF : Distributed Kalman Filtering with Low-cost Communications using the Sign of Innovations. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, number 2, pages 153–156, Toulouse.

Richard, C., Honeine, P., Snoussi, H., Ferrari, A., and Theys, C. (2010). Distributed learning with kernels in wireless sensor networks for physical phenomena modeling and tracking. In *Proc. 30th IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

Roecker, J. and McGillem, C. (1988). Comparison of two-sensor tracking methods based on state vector fusion and measurement fusion. *Aerospace and Electronic Systems, IEEE Transactions on*, 24(4):447 – 449.

Santi, P. (2005). *Topology Control in Wireless Ad Hoc and Sensor Networks*. Wiley, 1:st edition.

Sayed, A. H. (2012). Diffusion adaptation over networks. *arXiv preprint arXiv:1205.4220*.

Sayed, A. H., Tu, S.-Y., Chen, J., Zhao, X., and Towfic, Z. J. (2013). Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior. *Signal Processing Magazine, IEEE*, 30(3):155–171.

Scholkopf, B., Guyon, I., and Weston, J. (2003). Statistical learning and kernel methods in bioinformatics. *Nato Science Series Sub Series III Computer and Systems Sciences*, 183:1–21.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels*. MIT Press, Cambridge, MA.

Shanmuganthan, S., Ghobakhlou, A., Sallis, P., and Mastorakis, N. (2008). Sensors for modeling the effects of climate change on grapevine growth and wine quality. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*, number 12. World Scientific and Engineering Academy and Society.

Song, W., Wang, Y., Li, X., and Frieder, O. (2004). Localized algorithms for energy efficient topology in wireless ad hoc networks. *Proc. ACM MobiHoc*, pages 98–108.

Spanos, D., Olfati-Saber, R., and Murray, R. (2006). Dynamic consensus on mobile networks. In *Information Processing in IFAC World Congress*.

Speyer, J. (1979). Computation and transmission requirements for a decentralized linear-quadratic-gaussian control problem. *Automatic Control, IEEE Transactions on*, 24(2):266 – 269.

Swami, A., Zhao, Q., Hong, Y.-W., and Tong, L. (2007). *Wireless Sensor Networks: Signal Processing and Communications*. John Wiley & Sons.

Tamassia, R. and Tollis, I. G. (1989). Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234.

Tiberi, U., Fischione, C., Johansson, K. H., and Di Benedetto, M. (2013). Enegy-efficiency sampling of networked control systems over IEEE 802.15.4 wireless networks. *Automatica*, 13.

Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R., and Gill, C. (2003). Integrated Coverage and Connectivity for Energy Conservation in Wireless Sensor Networks. *Proc. ACM Int. Conf. Embedded Netw. Sens. Syst.*, pages 28–39.

Wieselthier, J., Nguyen, G., and Ephremides, A. (2000). On the construction of energy-efficient broadcast and multicast trees in wireless networks. *Proc. IEEE Infocom 00, Tel Aviv*, pages 585–594.

Willsky, A., Bello, M., Castanon, D., Levy, B., and Verghese, G. (1982). Combining and updating of local estimates and regional maps along sets of one-dimensional tracks. *Automatic Control, IEEE Transactions on*, 27(4):799 – 813.

Xiao, J.-J., Ribeiro, A., Luo, Z.-Q., and Giannakis, G. (2006). Distributed compression-estimation using wireless sensor networks. *Signal Processing Magazine, IEEE*, 23(4):27 – 41.

Xiao, L., Boyd, S., and Lall, S. (2005). A scheme for robust distributed sensor fusion based on average consensus. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 63 – 70.

Xie, S., Low, K. S., and Gunawan, E. (2014). An adaptive tuning algorithm for ieee 802.15.4 based network control systems. In *2014 IEEE ninth International Conference on Intelligent Sensor Network and Information Processing (ISSSNIP)*.

Zhang, H., Yang, L., Deng, W., and Guo, J. (2008). Handwritten chinese character recognition using local discriminant projection with prior information. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

Zhang, W., Branicky, M. S., and Phillips, S. M. (2001). Stability of Networked Control Systems. *IEEE Control Systems Magazine*.

Zhao, X. and Sayed, A. H. (2012). Performance limits for distributed estimation over lms adaptive networks. *Signal Processing, IEEE Transactions on*, 60(10):5107–5124.