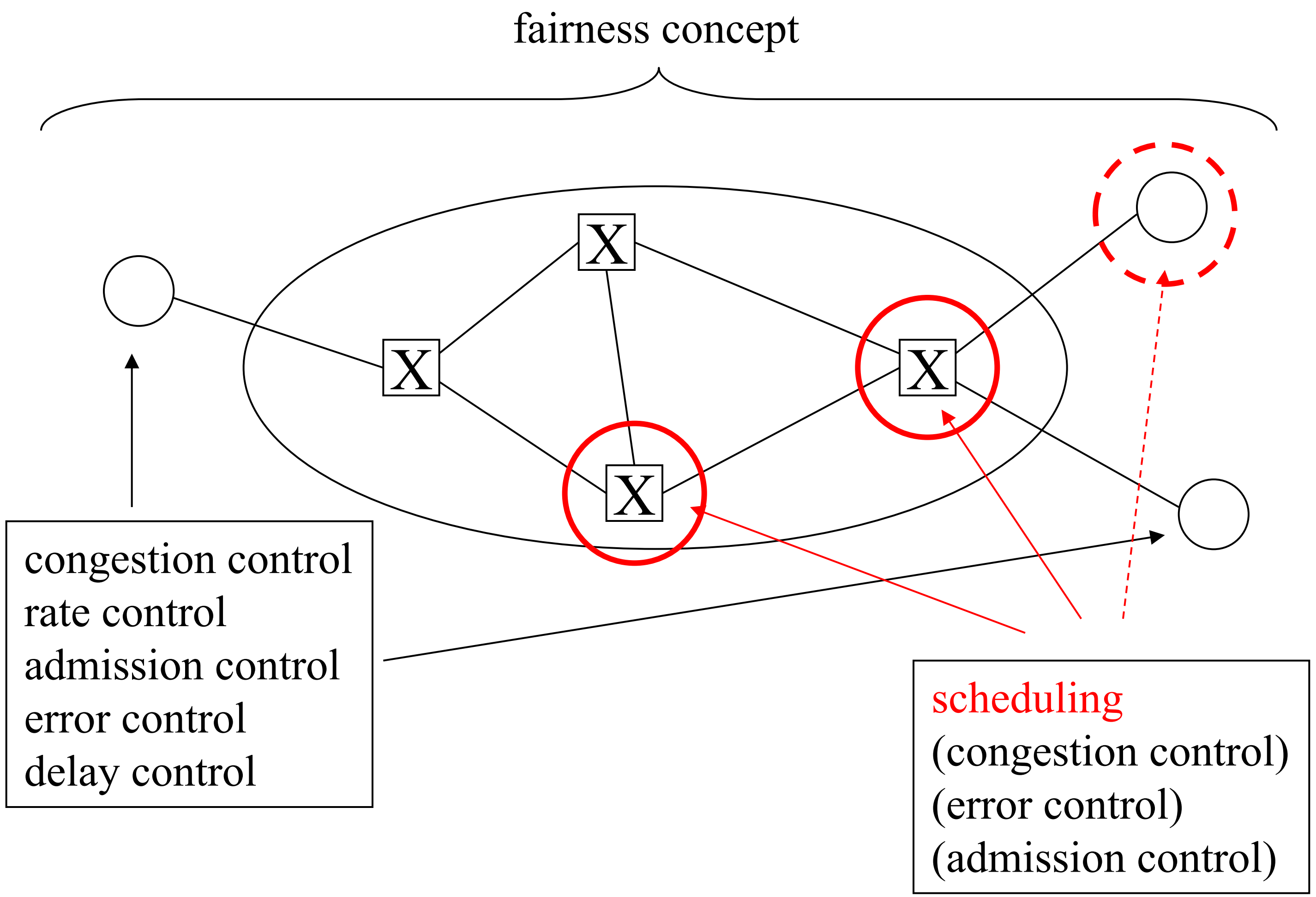
EP2210
Scheduling

- Lecture material:
  - Bertsekas, Gallager, 6.1.2.
  - MIT OpenCourseWare, 6.829
  - A. Parekh, R. Gallager, "A generalized Processor Sharing Approach to Flow Control - The Single Node Case," IEEE Infocom 1992

Scheduling


fairness concept
X
X
X
X
congestion control
rate control
admission control
error control
delay control
scheduling
(congestion control)
(error control)
(admission control)

Scheduling - Problem definition

- Scheduling happens at the routers (switches) – or at user nodes if there are many simultaneous connections
  - many flows transmitted simultaneously at an output link
  - packets waiting for transmission are buffered
- Question: which packet to send, and when?

- Simplest case: FIFO
  - packets of all flows stored in the same buffer in arrival order
  - first packet in the buffer transmitted when the previous transmission is complete
  - packet transmission in the order of packet arrival
  - packet arriving when buffer is full dropped

- Complex cases: separate queues for flows (or set of flows)
  - one of the first packets in the queues transmitted
  - according to some policy
  - needs separate queues and policy specific variable for each flow
    - PER FLOW STATE

Scheduling - Requirements

- Easy implementation
  - has to operate on a per packet basis at high speed routers
- Fair bandwidth allocation
  - for elastic (or best effort) traffic
  - all competing flows receive the some “fair” amount of resources
- Provide performance guarantees for flows or aggregates
  - service provisioning in the Internet (guaranteed service per flow)
  - guaranteed bandwidth for SLA, MPLS, VPN (guaranteed service for aggregates)
  - integrated services in mobile networks (UMTS, 4G)
- Performance metrics
  - throughput, delay, delay variation (jutter), packet loss probability
  - performance guarantees should be de-coupled
    (coupled e.g., high throughput -> low delay variation)

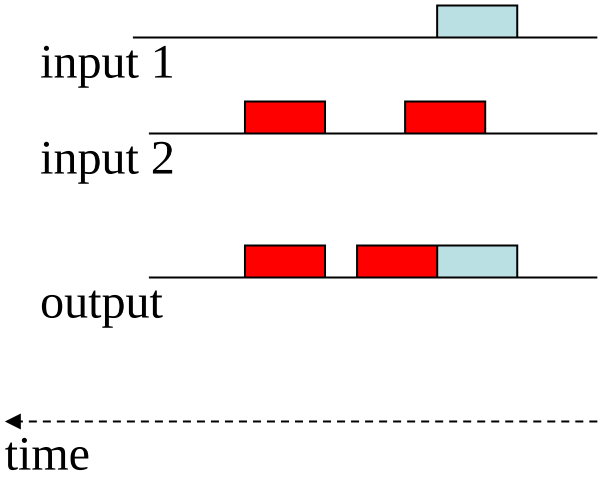
Scheduling – Implementation issues

- Scheduling discipline has to make a decision before each packet transmission – every few microseconds
- Decision complexity should increase slower than linearly with the number of flows scheduled
  - e.g., complexity of FIFO is 1
  - scheduling where all flows have to be compared scales linearly
- Information to be stored and managed should scale with the number of flows
  - e.g., with per flow state requirement it scales linearly (e.g., queue length or packet arrival time)

- Scheduling disciplines make different trade-off among the requirements on fairness, performance provisioning and complexity
  - e.g., FIFO has low complexity, but can not provide fair bandwidth share for flows

Scheduling classes

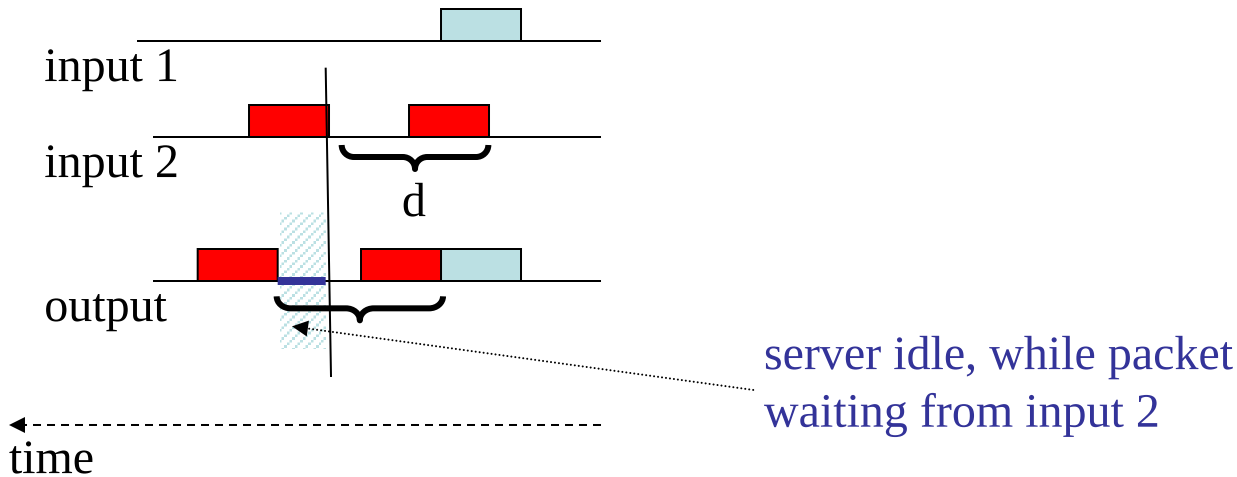
- Work-conserving
  - server (output link) is never idle when there is packet waiting



  - utilizes output bandwidth efficiently
  - burstiness of flows may increase → loss probability at the network nodes on the transmission path increases
  - latency variations at each switch → may disturb delay sensitive traffic

Scheduling classes

- Nonwork-conserving
  - add rate control for each flow
  - each packet assigned an eligibility time when it can be transmitted
    - e.g, based on minimum *d* gap between packets
  - server can be idle if no packet is eligible



  - burstiness and delay variations are controlled
  - some bandwidth is lost
  - can be useful for transmission with service guarantees

Scheduling for fairness

- The goal is to share the bandwidth among the flows in a “fair” way
  - fairness can be defined a number of ways (see lectures later)
  - here fairness is considered for one single link, not for the whole transmission path

- Max-min fairness
  - *Maximize* the *minimum* bandwidth provided to any flow not receiving all bandwidth it requests
  - E.g.: no maximum requirement, single node – the flows should receive the same bandwidth
  - Specific cases: weighted flows and maximum requirements

Max-min fairness

- *Maximize* the *minimum* bandwidth provided to any flow not receiving all bandwidth it requests

C: link capacity

B(t): set of flows with data to transmit at time t
(backlogged (saturated) flows)

n(t): number of backlogged flows at time t

$C_i(t)$: bandwidth received by flow i at time t

Case: without weights or max. requirements

$$C_i(t) = \frac{C}{n(t)}$$

Case: weights

w_i: relative weight of flow i

$$C_i(t) = \frac{w_i}{\sum_{j \in B(t)} w_j} C$$

Case: max. requirements

r_i: max. bandwidth requirement for flow i

$\alpha(t)$: fair share at time t

$$C_i(t) = \min(r_i, \alpha(t))$$

$$\alpha(t): \quad \sum_{j \in B(t)} \min(r_j, \alpha(t)) = C$$

Max-min fairness

C: link capacity

B(t): set of backlogged flows at time t

$C_i(t)$: bandwidth received by flow i at time t

Case: weights

w_i: relative weight of flow I

$$C_i(t) = \frac{w_i}{\sum_{j \in B(t)} w_j} C$$

Case: max. requirements

r_i: max. bandwidth requirement for flow I

$\alpha(t)$: fair share at time t

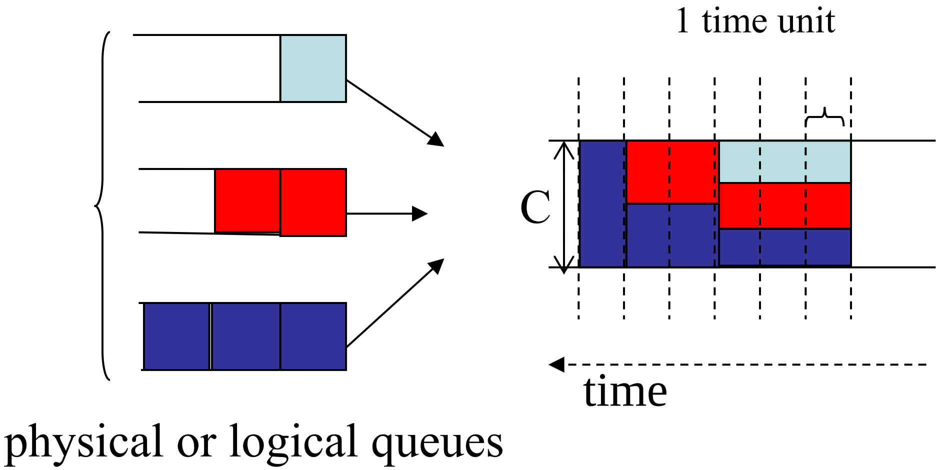
$$C_i(t) = \min(r_{i,} \alpha(t))$$

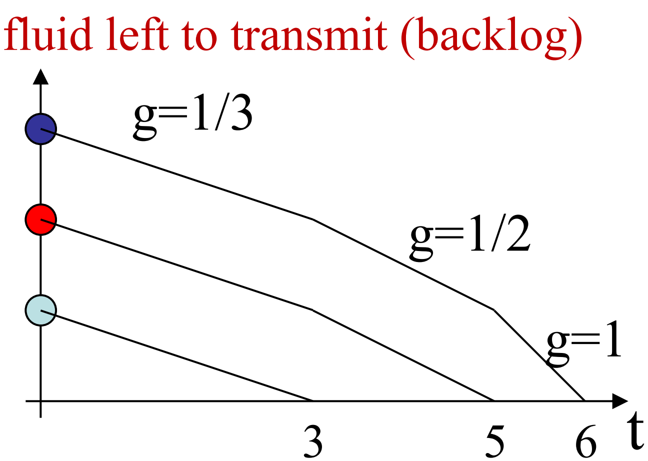
$$\alpha(t): \quad \sum_{j \in B(t)} \min(r_j, \alpha(t)) = C$$

- Calculate fair shares:
  - 3 backlogged (saturated) flows, equal weights, link capacity 10.
  - 3 backlogged flows, weights 1,2,2 link capacity 10
  - 4 backlogged flows, max requirements: 2, 3, 4, 5, link capacity 11.
  - 3 backlogged flows, rate requirements: 2,4,5, the link capacity is 11. What are the fair shares now?

Fair queuing-for max-min fairness

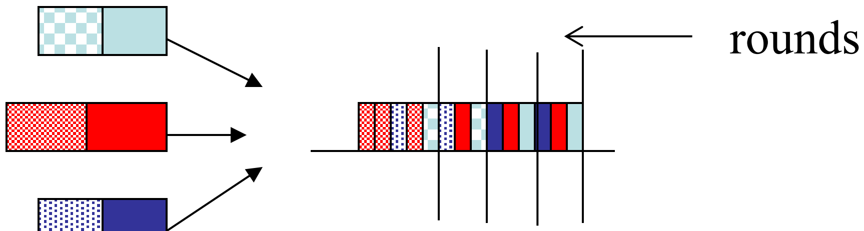
- Fluid approximation
  - fluid fair queuing (FFQ) or generalized processor sharing (GPS)
  - idealized policy to split bandwidth
  - assumption: dedicated buffer per flow
  - assumption: flows from backlogged queues served simultaneously (like fluid)
  - not implementable, used to evaluate real approaches
  - used for performance analysis if per packet performance is not interesting



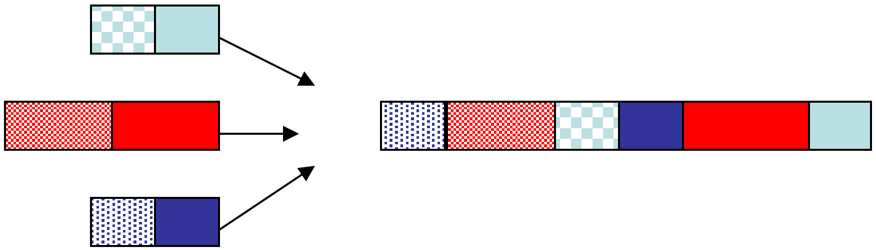


Packet-level Fair queuing

- How to realize GPS/FFQ?
- Bit-by-bit fair queuing
  - one bit from each backlogged queue in rounds (round robin) – still not possible to implement



- Packet-level fair queuing
  - one packet from each backlogged queue in rounds ???

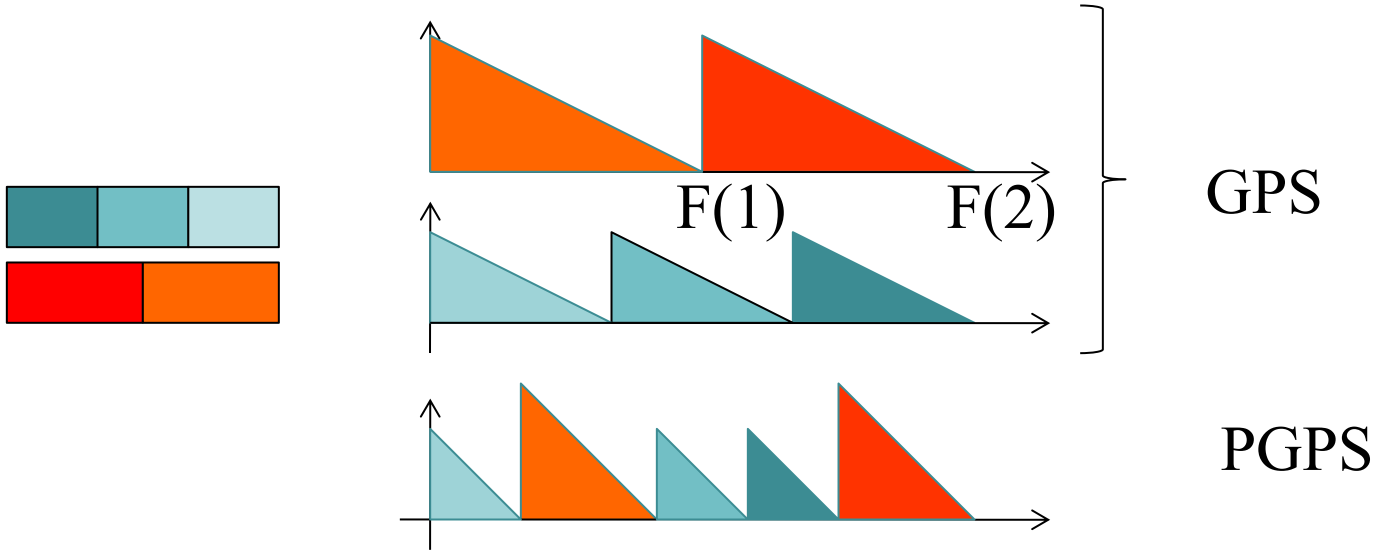


Flows with large packets get more bandwidth!

More sophisticated schemes required!

Packetized GPS (PGPS)

- How to realize GPS/FFQ?
- Try to mimic GPS
- Transmit packets that would arrive earliest with GPS
  - Finishing time (F(p))
- Quantify the difference between GPS and PGPS



Fair queuing – group work

- Packet-by-packet GPS (PGPS)
- Compare GPS (fluid) and PGPS (packetized) in the following scenarios – draw diagrams "backlogged traffic per flow vs. time".
- Consider one packet in each queue. C=1 unit/sec

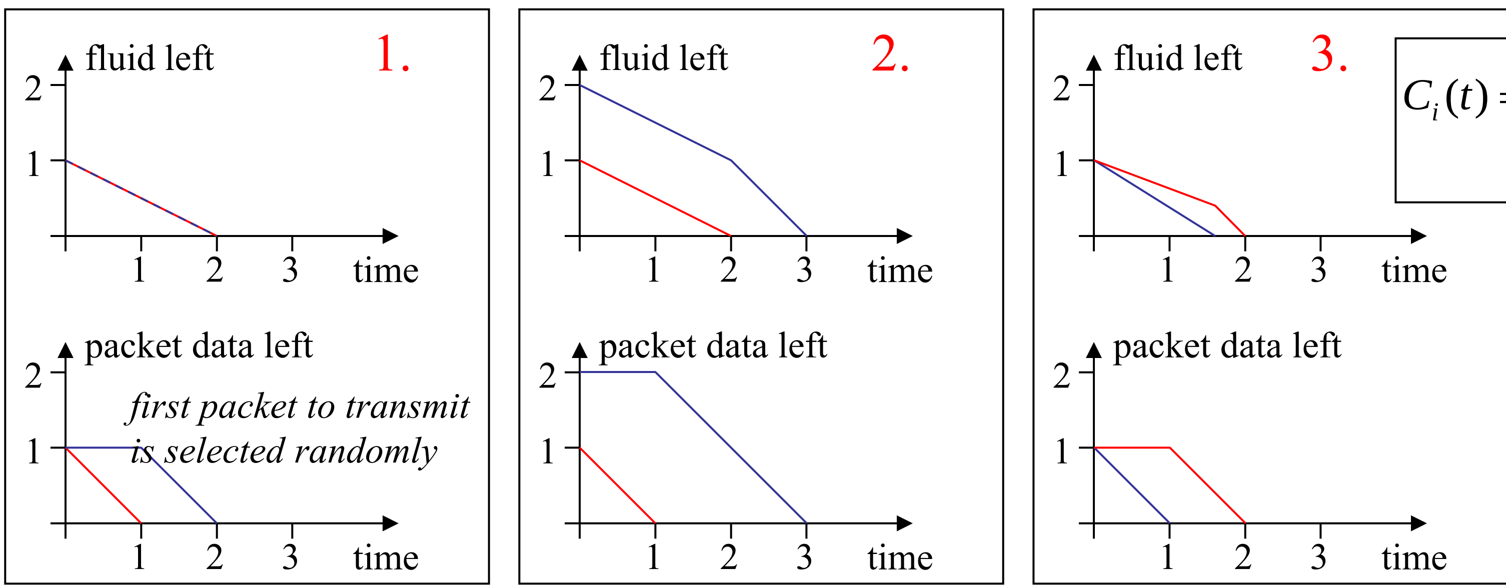
1. Two flows, equal size packets, same weight, L1=L2=1 unit
2. Two flows, different size packets, same weight L1=1, L2=2 units
3. Two flows, same packet size, different weight, L1=L2=1 unit, w1=1, w2=2

$$C_i(t) = \frac{w_i}{\sum_{j \in B(t)} w_j} C$$

Fair queuing – group work

- Compare GPS (fluid) and PGPS (packetized) in the following scenarios – draw diagrams “backlogged traffic per flow vs. time”.
- Consider one packet in each queue. C=1 unit/sec

1. Two flows, equal size packets, same weight, L1=L2=1 unit
2. Two flows, different size packets, same weight L1=1, L2=2 units
3. Two flows, same packet size, different weight, L1=L2=1 unit, w1=1, w2=2



$$C_i(t) = \frac{w_i}{\sum_{j \in B(t)} w_j} C$$

Scheduling summary

- Scheduling:
  - At the network nodes and at the edge
  - To provide quality guarantees or fairness
  - Work-conserving and non-work-conserving
- Max-min fairness in a single link, with weights and max. rate requirement
- GPS for max-min fairness in a fluid model
- PGPS (or WFQ) in the packetized version
  - Schedule according to finish time in GPS
  - Guaranteed performance compared to GPS
- Next lecture: PGPS in detail, work-conserving and non-work-conserving scheduling

Reading assignment

- A. Parekh, R. Gallager, “A Generalized Processor Sharing Approach to Flow Control - The Single Node Case,” IEEE Transaction on Networking, 1993, Vol.1, No.3.
  - Read from I to III-before part A

- H. Zhang, “Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks,” Proceedings of the IEEE, Oct, 1995, pp. 1374-1396
  - Read sections I, II, and III.

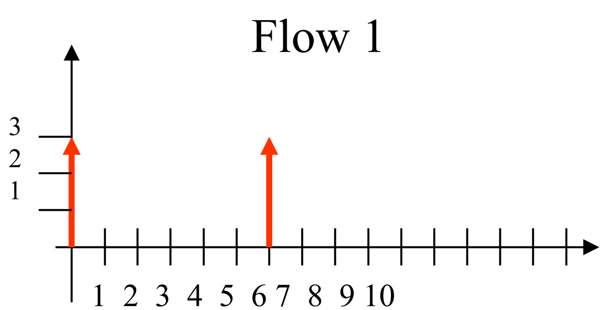
Lecture plan

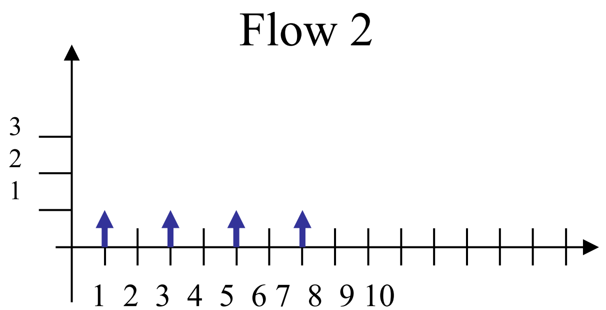
- GPS versus PGPS student presentation
- GPS under random arrivals, the M/M/1-PS queue
- Effect of scheduling over multiple hops – the Zhang Paper

Scheduling - GPS, PGPS

- Consider two flows sharing a link. Packet arrivals and sizes are shown on the figure. Draw a figure explaining how the packets are served with GPS and give the finishing time of each packet. (arrivals: t=0,6 and t=1,3,5,7)

- How are the same packets transmitted under PGPS (packet based GPS)?





Processor sharing queue

- The performance of GPS (single link or single resource) under stochastic request arrival.

- Recall: for FIFO service, Poisson arrivals, Exp service time
  - FIFO, single server – M/M/1
  - FIFO, multiple servers – M/M/m
  - FIFO, infinite servers – M/M/inf

- Question: how can we model the GPS service?
  - Assume Poisson arrivals
  - Assume Exponential service time

Processor sharing queue

- The performance of GPS (single link or single resource) under stochastic request arrival. Fluid model.

- Single server (single link, transmission medium or resource)
- The capacity of the server equally shared by the requests
  - if there are n requests, each receives service at a rate 1/n
  - customers do not have to wait at all, service starts as the customer arrives (there is no queue…)
- M/M/1-PS
  - Poisson customer arrival process (λ)
  - Service demand (job size) is exponential in the sense, that if the customer got all the service capacity, then the service time would be Exp(μ) (models e.g., exponential file size)
  - Note: if the number of requests is higher, a request stays in the server for a longer time.

Processor sharing queue

- M/M/1-PS
  - Poisson customer arrival process (λ)
  - service demand (job size) is exponential in the sense, that if the customer got all the service capacity, then the service time would be Exp(μ)
- Draw the Markov chain
- Compare it to the M/M/1-FIFO queue.

- Consequently, p, E[N], and E[T] is the same as M/M/1-FIFO

$$p(n) = (1-\lambda/\mu)(\lambda/\mu)^n, \quad E[N] = \frac{\lambda/\mu}{1-\lambda/\mu}, \quad E[T] = \frac{E[N]}{\lambda} = \frac{1/\mu}{1-\lambda/\mu}$$

- Moreover, the average results are the same for M/G/1-PS – average measures are insensitive to the service time distribution

Processor sharing queue

- M/M/1-PS example
- WLAN access point (10Mbit/s) is shared for large file transfer. File transfers are initiated randomly by a large population, the file sizes are considered to be exponential. The average file size is 1MByte.
- We assume that the medium access control does not waste capacity

- How much time does it take in average to download a file, if noone else is downloading?

Assume, file downloads are initiated with a rate of 0.5 per second

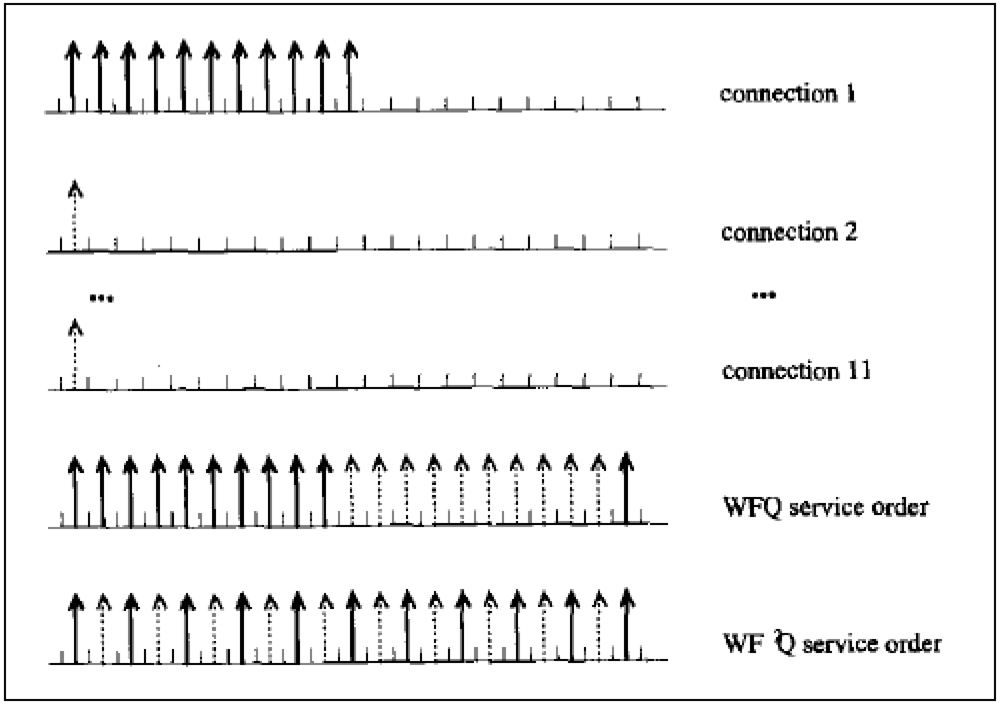
- Give the MC of the system
- What is the probability that the network is empty?
- What is the mean number of concurrent downloads and time to download a file?
- Express the probability that the instantaneous rate is less than 1Mbit/s?...

Back to scheduling algorithms

- Introduction to the Hui Zhang paper
- Scheduling for guaranteed services – all flows have some limited requirements (average rate, traffic envelope …)
- Work-conserving: WFQ, WFFQ
- Non-work-conserving: Jitter EDD, Stop-and-Go

Work conserving: WFQ and WFFQ

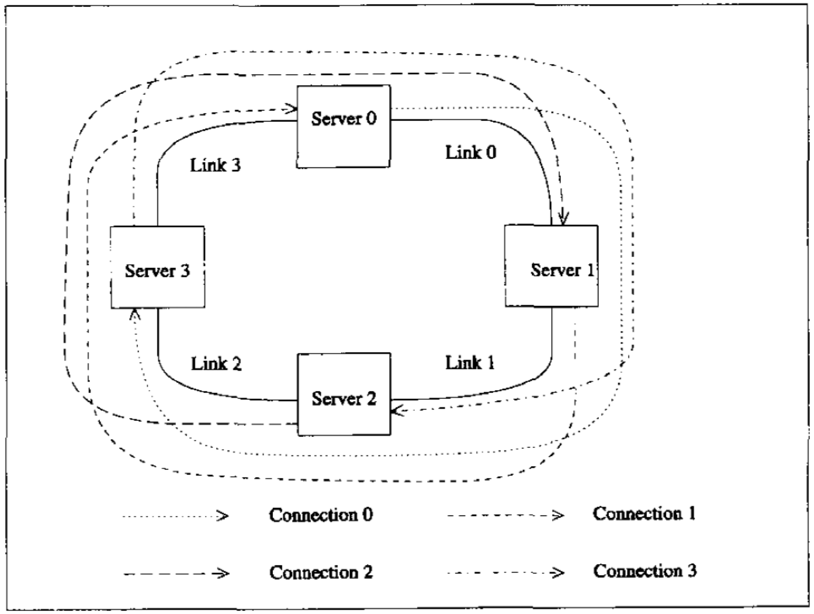
- Weighted Fair Queuing (same as PGPS)
  - Orders packets according to finishing times in FFQ (fluid fair…)
  - Can schedule packets too much ahead of FFQ
- WFFQ Worst-case fair weighted fair …
  - Considers only the packets that have started service under FFQ
  - Leads to less bursty traffic



Guaranteed rate:
Connection 1: 0.5
Connections 2-11: 0.05

Work conserving: troubles

- Increasing burstiness
- Traffic characterization and stability region
  - Set of equations
- E.g., feedback network
  - Set of equations may be unsolvable…
  - The network can become unstable



Non-work-conserving: jitter-EDD, Stop-and-Go

- Jitter-Earliest-Due-Date
  - Keep jitter limited
  - While utilize free link under some constraints

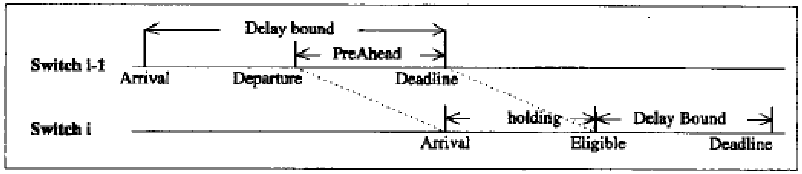


Fig. 10. Packet service in jitter-EDD.

- Stop-and-Go
  - Window based control
  - Received in one window is transmitted in one window (with some delay…)

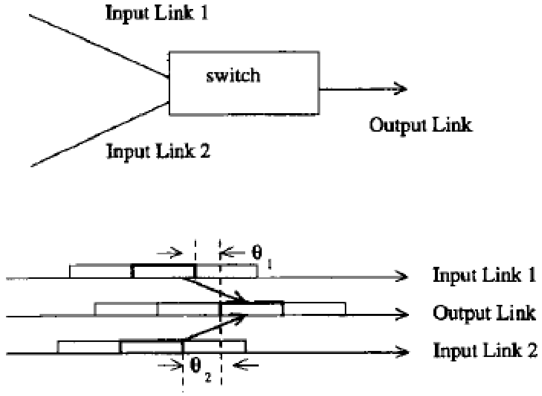


Fig. 11. Synchronization between input and output links in stop-and-go.

Performance comparison

Table 2 End-to-End Delay, Bound Delay, Delay-Jitter, and Buffer Space Requirements

	traffic constraint	end-to-end delay bound	end-to-end delay-jitter bound	buffer space at h^{th} switch
D-EDD	$b_j(\cdot)$	$\sum_{i=1}^{n} d_{i,j}$	$\sum_{i=1}^{n} d_{i,j}$	$b_j(\sum_{i=1}^{h} d_{i,j})$
FFQ	(σ_j, ρ_j)	$\frac{\sigma_j}{r_j}$	$\frac{\sigma_j}{r_j}$	σ_j
VC	(σ_j, ρ_j)	$\frac{\sigma_j + nL_{max}}{r_j} + \sum_{i=1}^{n} \frac{L_{max}}{C_i}$	$\frac{\sigma_j + nL_{max}}{r_j}$	$\sigma_j + hL_{max}$
WFQ & WF^2Q	(σ_j, ρ_j)	$\frac{\sigma_j + nL_{max}}{r_j} + \sum_{i=1}^{n} \frac{L_{max}}{C_i}$	$\frac{\sigma_j + nL_{max}}{r_j}$	$\sigma_j + hL_{max}$
SCFQ	(σ_j, ρ_j)	$\frac{\sigma_j + nL_{max}}{r_j} + \sum_{i=1}^{n} K_i \frac{L_{max}}{C_i}$	$\frac{\sigma_j + nL_{max}}{r_j} + \sum_{i=1}^{n} (K_i - 1) \frac{L_{max}}{C_i}$	$\sigma_j + hL_{max}$

Table 4 End-to-End Delay, Delay Jitter, and Buffer Space Requirement for Nonwork-Conserving Disciplines

	traffic constraint	end-to-end delay bound	end-to-end delay-jitter bound	buffer space at h^th switch
Stop-and-Go	(r_j, T_j)	$nT_j + \sum_{i=1}^{n} \theta_i$	T_j	$r_j(2T_j + \theta_i)$
HRR	(r_j, T_j)	$2nT_j$	$2nT_j$	$2r_jT_j$
Rate-Controlled Servers with $b^*(\cdot)$ RJ regulators	$b_j(\cdot)$	$D(b_j, b^*) + \sum_{i=1}^{n} d_{i,j}$	$D(b_j, b^*) + \sum_{i=1}^{n} d_{i,j}$	$\sigma_j + b^*(d_{1,j})$ for 1st switch; $b^*(d_{i-1,j} + d_{i,j})$ for j^{th} switch $j > 1$
Rate-Controlled Servers with $b^*(\cdot)$ RJ regulator for 1st switch and DJ regulators for other switches	$b_j(\cdot)$	$D(b_j, b^*) + \sum_{i=1}^{n} d_{i,j}$	$D(b_j, b^*) + d_{n,j}$	$\sigma_j + b^*(d_{1,j})$ for 1st switch; $b^*(d_{i-1,j} + d_{i,j})$ for j^{th} switch $j > 1$

Scheduling - summary

- Scheduling: local algorithms to decide which packet to transmit
- Scheduling for fairness
  - Generalized processor sharing, fluid fair queueing, M/M/1-PS
  - Packetized versions
- Scheduling for performance guarantees
  - Work-conserving examples: WFQ, WFFQ
  - Non-work-conserving examples: Jitter-EDD, S&G
  - Performance evaluation:
    - Delay bound, jitter bound, buffer space
    - Dependence on number of hops
    - Correlated performance (e.g., rate vs jitter)

- Material for test: everything discussed in class
- Material for home assignment: more reading….