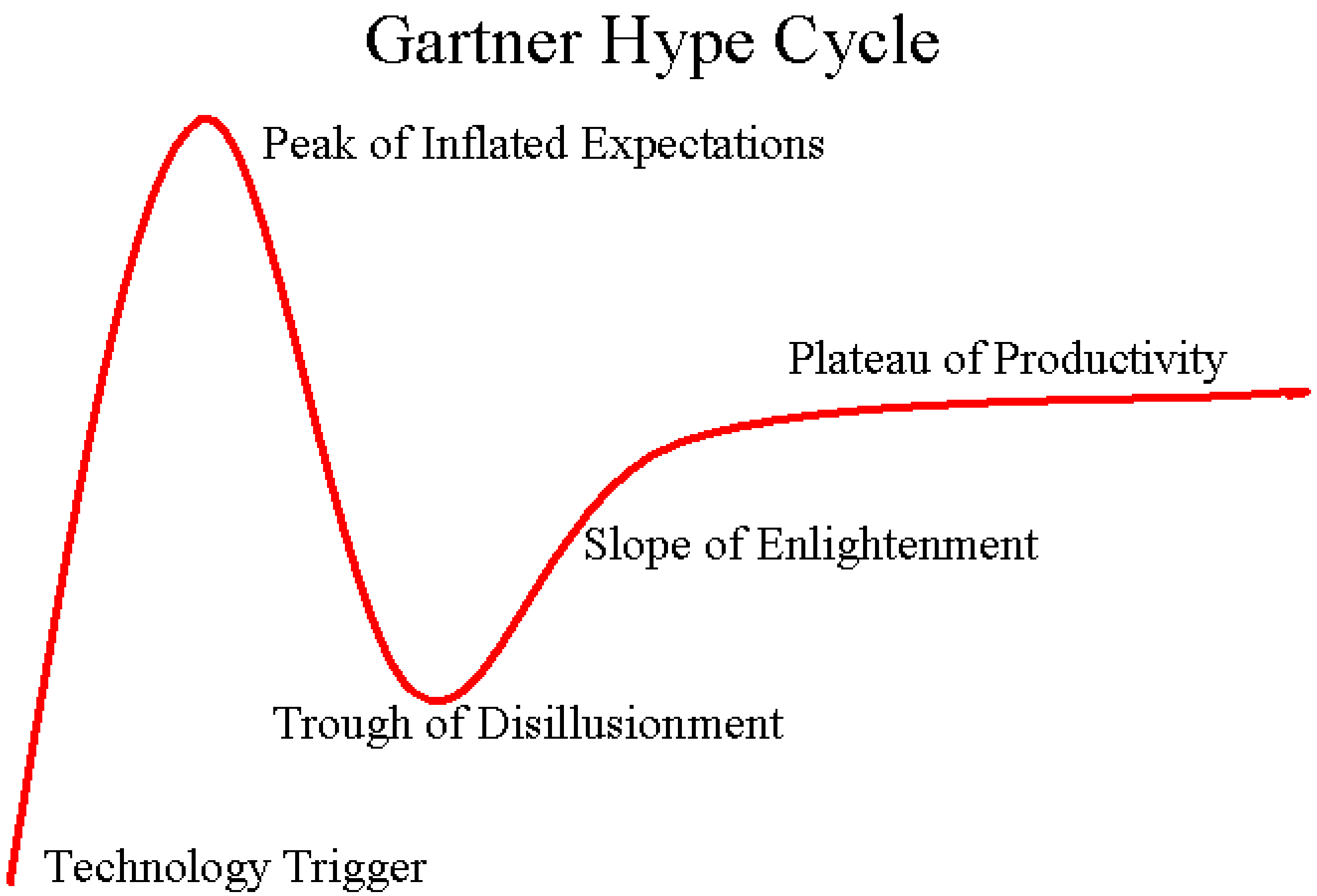
Peer-to-Peer Systems and ~~GRID Computing~~ Cloud Computing ID2210

Jim Dowling (jdowling@kth.se)
Salman Niazi (salman@sics.se)
Amir H. Payberah (amir@sics.se)

Some slides by Šarūnas Girdzijauskas

Transformative Technologies



Course Objectives

Learning Objectives

- To understand and apply the main concepts and principles from large-scale dynamic decentralized systems.

- Implement and evaluate peer-to-peer algorithms in a simulation environment.

- How to read, review and present a scientific paper.

- To understand the main concepts and principles from cloud computing when building a distributed system.

Topics of Study

- Fundamental results in large-scale distributed algorithms.
- Overview of peer-to-peer systems, algorithms, and applications.
- Study of Structured Overlay Networks (SONs).
- Gossip and Epidemic Overlay Networks.
- Content and Streaming Distribution Networks.
- Introduction to Cloud Computing and Hadoop
- Lectures by companies in P2P and Cloud Computing space

Material

- Mainly based on research papers.
- You will find all the material on the course web page:
  http://www.ict.kth.se/courses/ID2210/
- Assignments will be handled using Bilda:
  http://bilda.kth.se

Course Requirements

- Reading assignments: review 5 papers (20 points)
  - R1-R4: Peer-to-Peer Papers (4 each)
  - R5: Cloud-Computing (4)
- Quizzes (24 points)
  - Quiz 1 (12)
  - Quiz 2 (12)
- Presentation (pass/fail – 16 points)
  - presentation
  - attendance of two other sessions
- One home assignment (40 points)
  - Two consultation sessions
  - 10 pt per week late. Minimum 10 point deduction for late submission.
  - You must pass the home assignment to pass the course.

Reading Assignments

- You should write 5 review reports, each one at most one page.
- You will be given a template for this task.
- For each paper you should
  - Identify and motivate the problem
  - Pinpoint the main contributions
  - Identify positive aspects of the solution/paper
  - Identify weak points of the paper
- For each paper, you might be given some questions to answer in a separate page.
- Hard deadlines!

Lab Assignment

- Implement a peer-to-peer system in Kompics.
- Build this system step by step according to the specifications.
- Evaluate in simulation, the performance and properties of the implemented system.
- Report your results in a document.
- Passing this task is mandatory for you to pass the course. If you miss the deadline, you will be deducted 10 points per week late (with a minimum of 10 points being deducted).
- Grading scheme:
  - A: 90 – 100
  - B: 80 – 89
  - C: 70 – 79
  - D: 60 – 69
  - E: 50 – 59
  - F: < 50

Quiz and Presentation

Quiz

- In the quiz we will ask questions based on the lectures notes and the corresponding papers.

Presentation

- You give a 15 minutes talk on a scientific paper.

- The list of papers will be available in the course web page.

- You are free to choose any other paper, but it should be confirmed by the course staff.

Final Grade

- You should work in groups of 2 persons for reading assignments, lab assignments and final presentation.
  - Report your group information to me by email by the end of the week.
  - After this deadline, we will decide on the group members.

- Final grade is determined by the sum of the points you collect, according to the following scheme:
  - A: 90 – 100
  - B: 80 – 89
  - C: 70 – 79
  - D: 60 – 69
  - E: 50 – 59
  - F: < 50

Discussion Forum

- Use Bilda for course discussion.

- Course Website with lecture notes.
  https://www.kth.se/social/course/ID2210/

Teachers

- Course responsible
  - Jim Dowling (jdowling@kth.se)
- Teaching assistants
  - Salman Niazi (salman@sics.se)
- Guest Lecturers
  - Amir H. Payberah
  - Roberto Roverso
  - Gunnar Kreiz

Course Overview

BitTorrent™

napster.

Spotify™

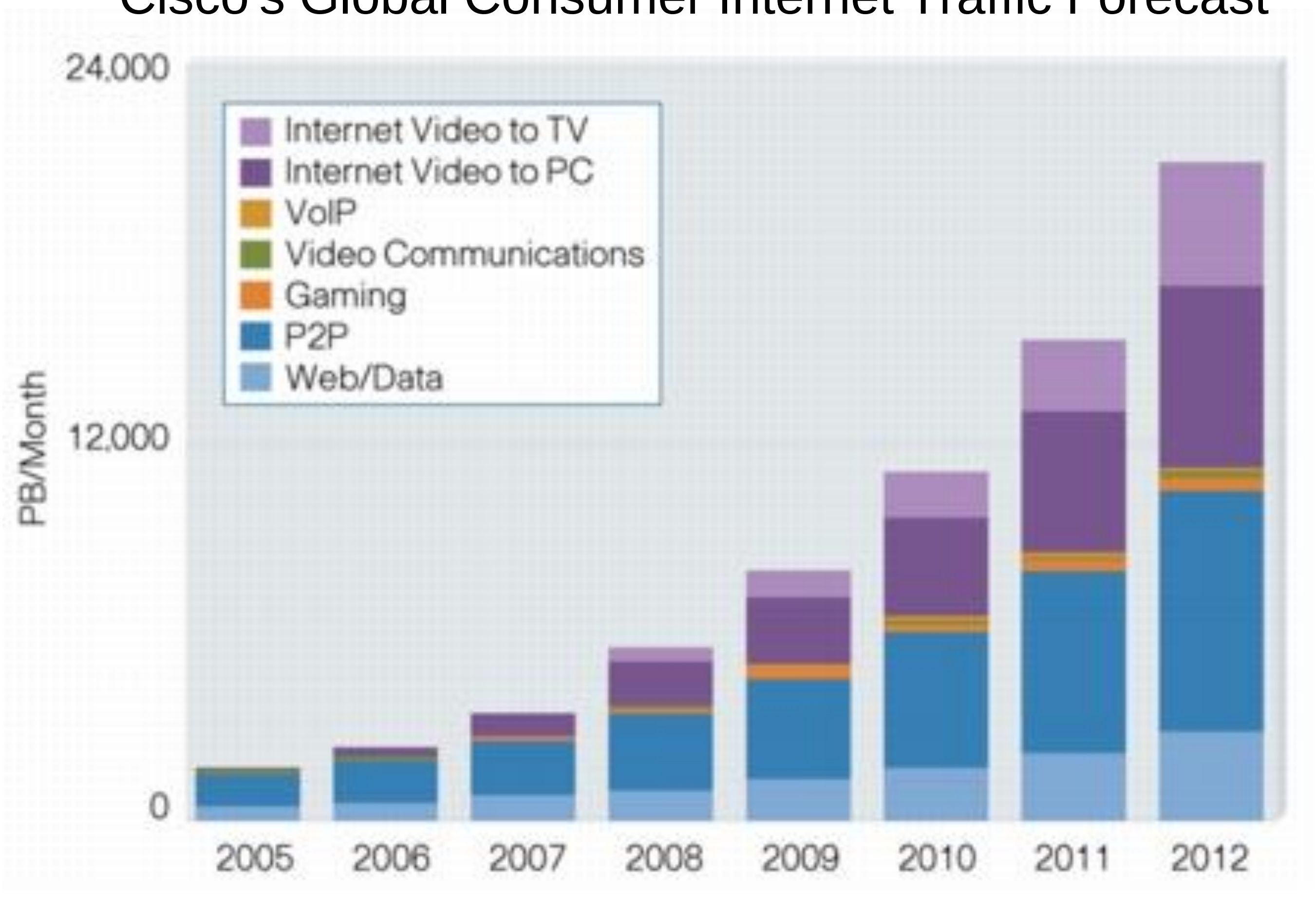
LimeWire

KaZaA

voddler

P2P? Why Should We Care?

Cisco's Global Consumer Internet Traffic Forecast



Outline

- What is P2P?
- P2P overlay types
  - Centralized
    - Napster
  - Unstructured: Flooding-Based systems, Gossip-based systems
    - Gnutella, Cyclon, etc.
  - Super-Peer networks: heterogeneous, hybrid systems
    - Kazaa
  - Structured: Distributed Hash Tables (DHT)
    - Chord, Dynamo, Kademlia, etc.

What is P2P Computing? (1/3)

- Oram (First book on P2P): P2P is a class of applications, that
  - Takes advantage of resources – (storage, cpu, etc,..) – available at the edges of the Internet.
  - Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have significant or total autonomy from central servers.

What is P2P Computing? (2/3)

- P2P Working Group (A Standardization Effort): P2P computing is
  - The sharing of computer resources and services by direct exchange between systems.
  - Peer-to-peer computing takes advantage of existing computing power and networking connectivity, allowing economical clients to leverage their collective power to benefit the entire enterprise.

What is P2P Computing? (3/3)

- Our view: P2P computing is distributed computing with the following desirable properties:
  - Resource Sharing
  - Dual client/server role
  - Decentralization/Autonomy
  - Scalability
  - Robustness/Self-Organization

P2P Research Issues

- Discovery
  - Where are things?
- Content Distribution
  - How fast can we get things?
- Connectivity
  - How do we address and connect to nodes behind NATs/Firewalls?
- Communications
  - How can we achieve reliable communication over unreliable links.
- Security
- Anonymity

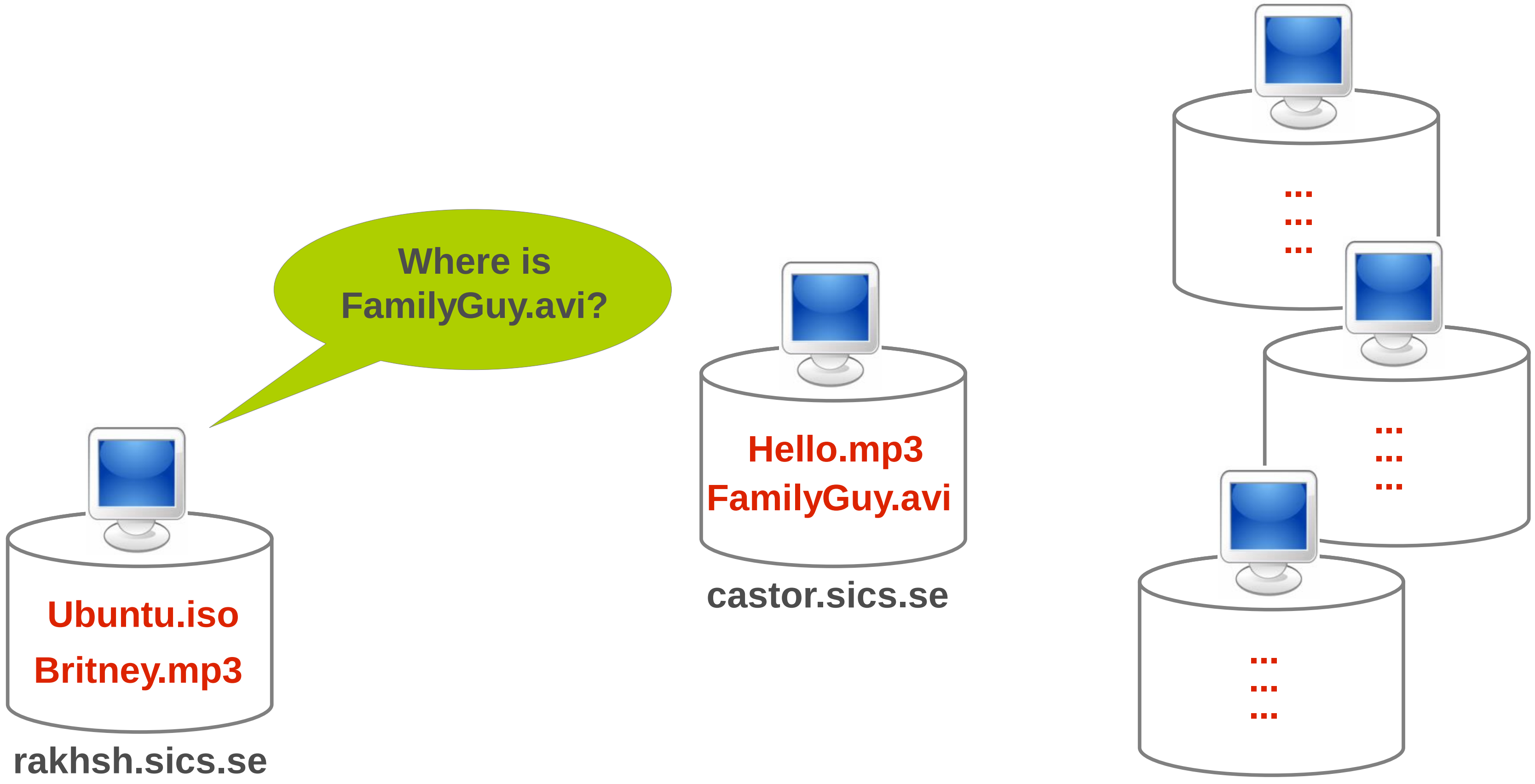
Let us see how did it all start ...

- Some users store data items on their machines.
- Other users are interested in this data.
- Problem: ?

Let us see how did it all start ...

- Some users store data items on their machines.
- Other users are interested in this data.
- Problem:
- How does a user know which other user(s) in the world have the data item(s) that s/he desires?

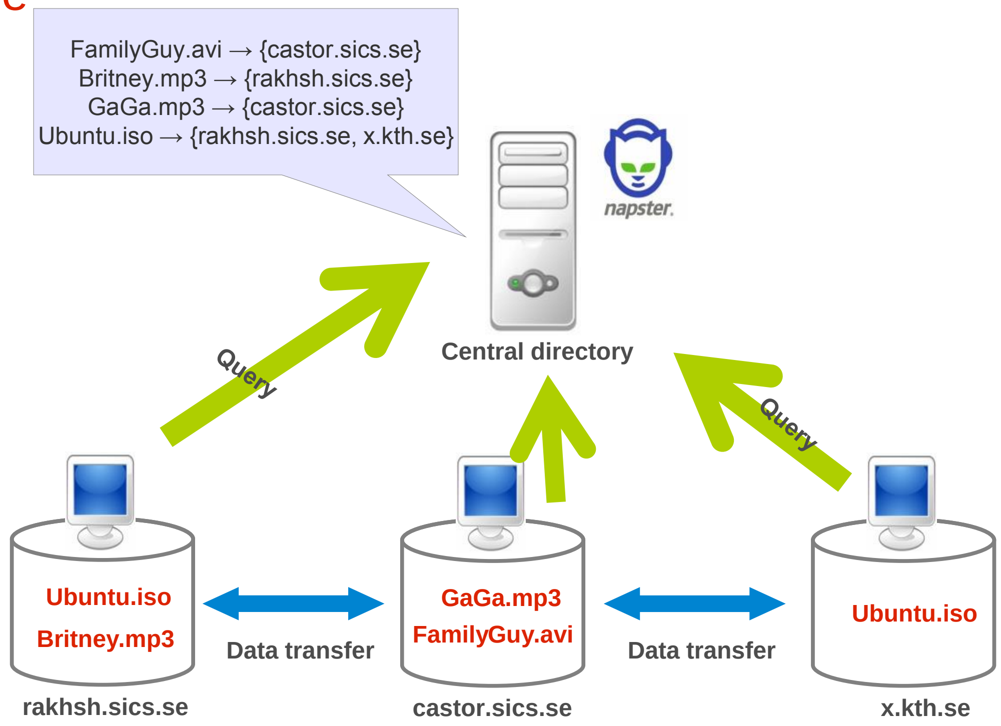
Let us see how did it all start ...



A centralized solution

A centralized solution: Napster (1)

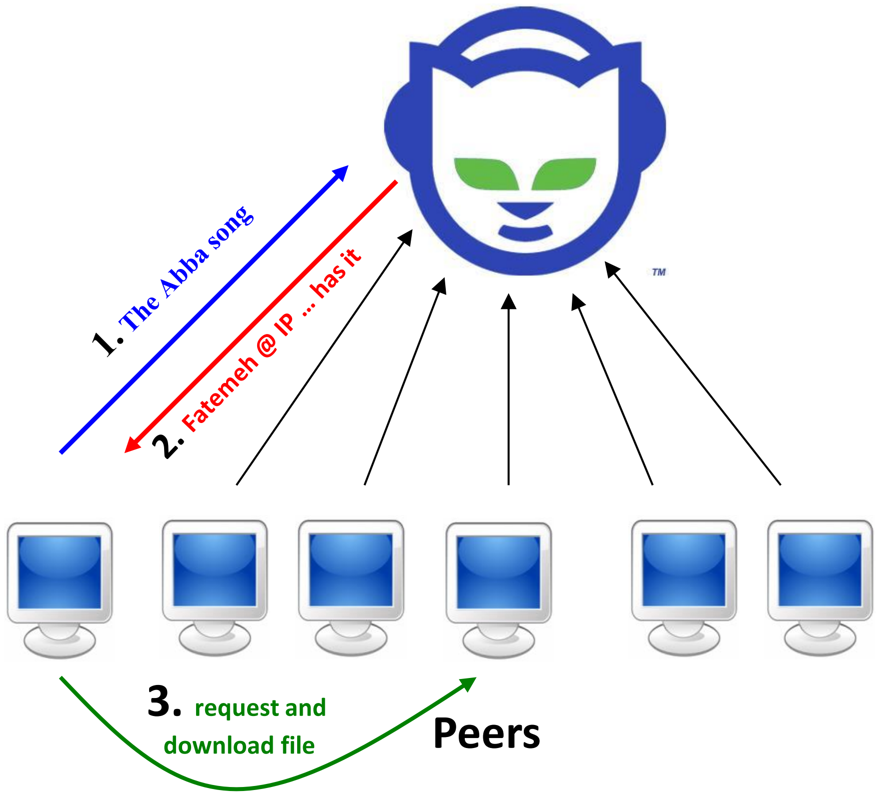
- Central Directory + Distributed Storage



- Basic Operations:
- Join
  - Connect to the central server (Napster)
- Share (Publish/Insert)
  - Inform the server about what you have
- Leave/Fail
  - Simply disconnect
  - Server detects failure, removes your data from the directory
- Search (Query)
  - Ask the central server and it returns a list of hits
- Download
  - Directly download from other nodes using the hits provided by the server

A centralized solution: Napster (2)

- Stats from Feb 2001:
  - 1.57 million users
  - 10 TeraByte of data
  - 220 songs per user
- Strengths: simple & efficient
  - Resource Sharing
    - Every node “pays” for its participation by providing access to its resources
    - Every participating node acts as both a client and a server :P2P
  - Global information system without huge investment
  - Decentralization for the resource-intensive functionalities: storage & bandwidth
- Weaknesses
  - Centralization for search: O(N) state in the server
  - Single point of administration: Centralized design made it easier to shut it down



An unstructured solution

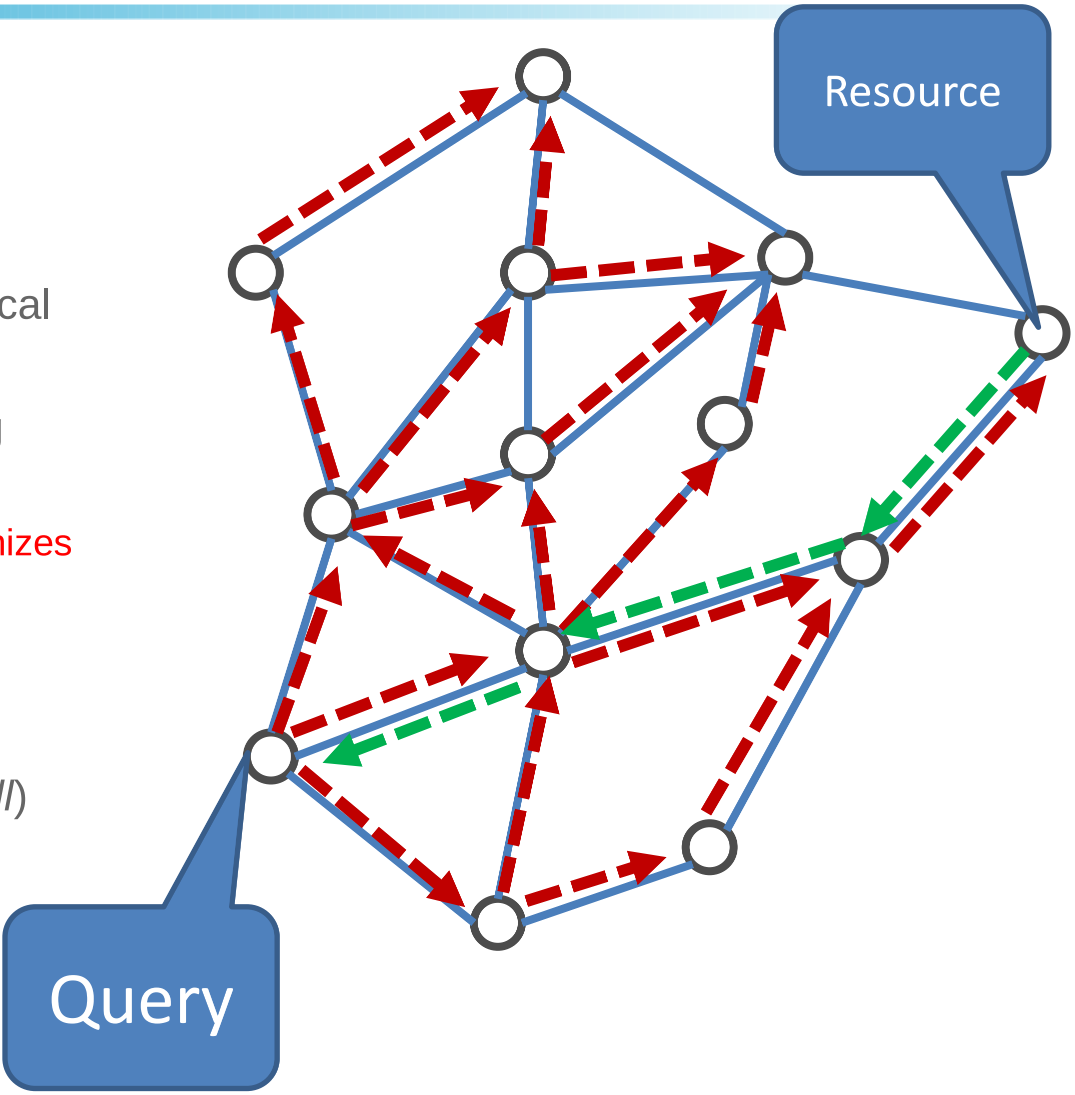
Unstructured P2P Networks

- Napster: Central Directory + Distributed Storage

- Gnutella: Complete decentralization
  - The first client was developed by Justin Frankel and Tom Pepper of Nullsoft in early 2000, soon after the company's acquisition by AOL.
    - Originally intended to share recipes
  - On March 14, the program was made available for download on Nullsoft's servers.
    - The event was prematurely announced on Slashdot, and thousands downloaded the program that day.
    - The source code was to be released later, under the GNU General Public License (GPL).
  - The next day, AOL stopped the availability of the program over legal concerns and restrained Nullsoft from doing any further work on the project.
  - This did not stop Gnutella; after a few days, the protocol had been reverse engineered
    - compatible free and open source clones began to appear.

Unstructured P2P Networks

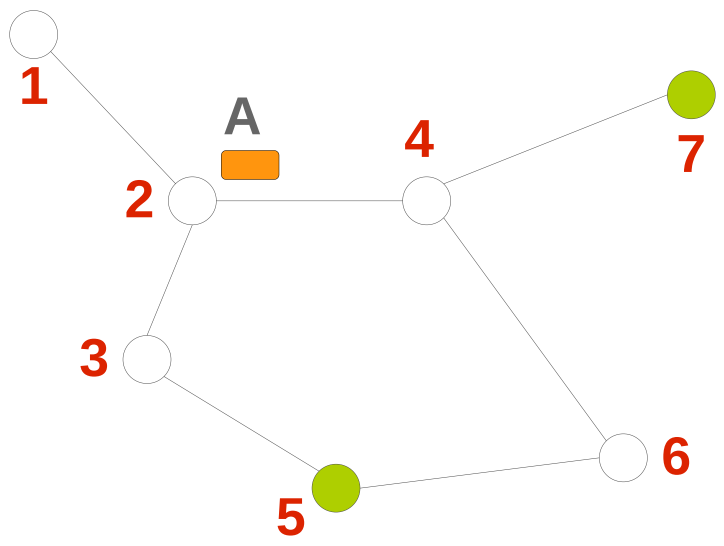
- How does it work?
  - Nodes establish some local connections
    - Ping/pong messages
  - Search requests (queries) are forwarded/flooded along these local connections, and answered by suitable peers
  - Local cooperation of participating nodes
    - Emerging global properties
    - The network/topology self-organizes
- Advantages
  - No single point of failure
  - No investments or administration
- Drawbacks
  - High search overheads
  - No guarantees for success (*recall*)
    - Flooding with TTL
  - No enforcement of user behavior
    - Anti-social behaviours possible
      - free-riding
      - polluting
      - …



Gnutella Protocol Messages

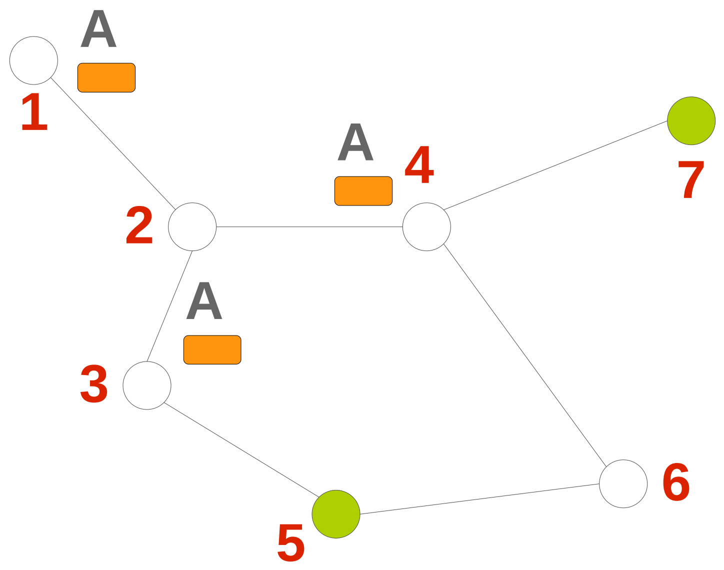
- Broadcast Messages
  - Ping: initiating message ("I’m here") for overlay maintenance
  - Query: search pattern and TTL (time-to-live)
- Back-Propagated Messages
  - Pong: reply to a ping, contains information about the peer
  - Query Hit: contains information about the computer that has the requested file
- Node-to-Node Messages
  - GET: return the requested file
  - PUSH: push the file to the requester node

Gnutella Search Mechanism



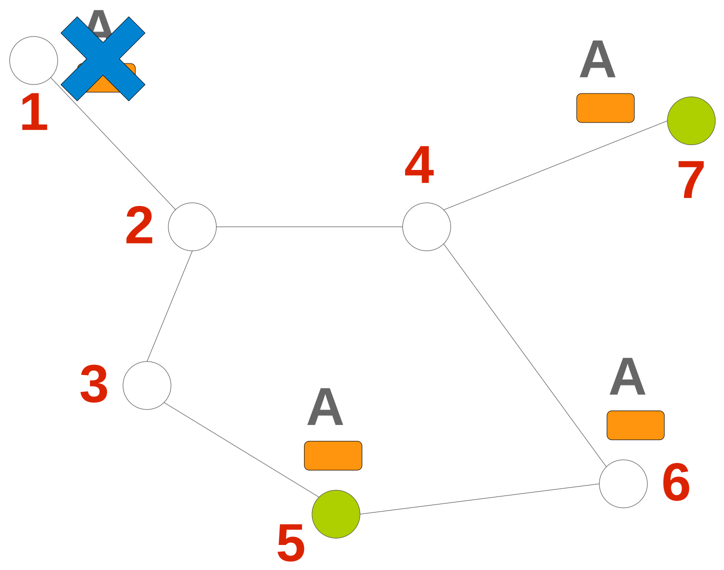
- Node 2 initiates search for file A

Gnutella Search Mechanism



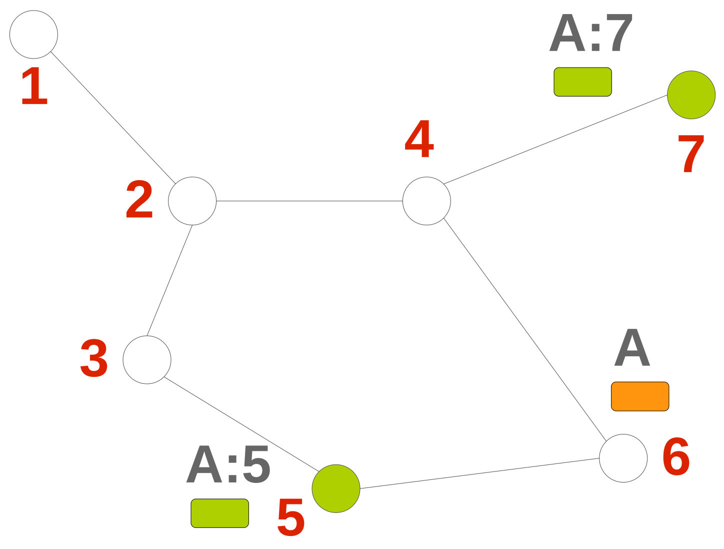
- Node 2 initiates search for file A
- Sends message to all neighbours

Gnutella Search Mechanism



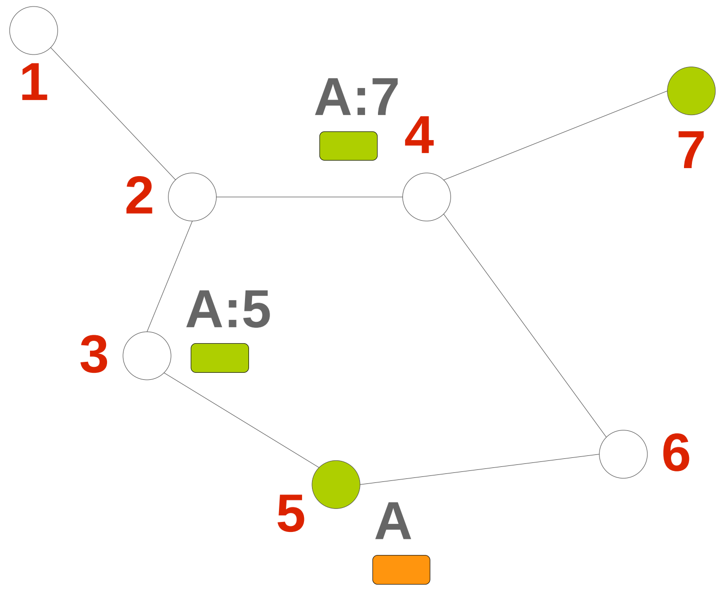
- Node 2 initiates search for file A
- Sends message to all neighbours
- Neighbours 3,4 forward message

Gnutella Search Mechanism



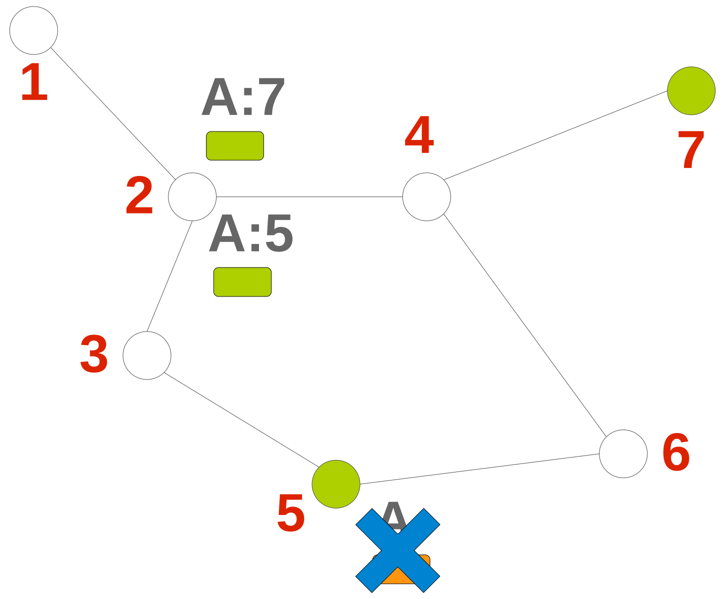
- Node 2 initiates search for file A
- Sends message to all neighbours
- Neighbours 3,4 forward message
- Nodes 5, 7 that have file A initiate a reply message

Gnutella Search Mechanism



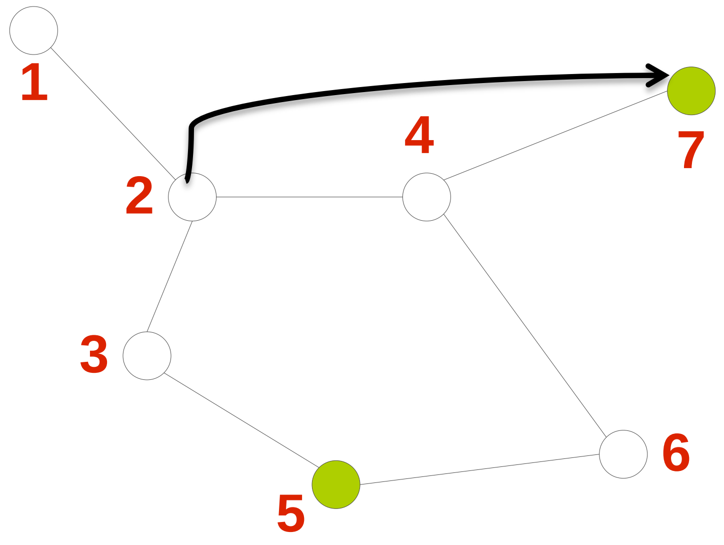
- Node 2 initiates search for file A
- Sends message to all neighbours
- Neighbours 3,4 forward message
- Nodes 5, 7 that have file A initiate a reply message
- Query reply message is back propagated

Gnutella Search Mechanism



- Node 2 initiates search for file A
- Sends message to all neighbours
- Neighbours 3,4 forward message
- Nodes 5, 7 that have file A initiate a reply message
- Query reply message is back propagated

Gnutella Search Mechanism



- Node 2 initiates search for file A
- Sends message to all neighbours
- Neighbours 3,4 forward message
- Nodes 5, 7 that have file A initiate a reply message
- Query reply message is back propagated
- Node 2 directly connects to node 7 and downloads file A

Cost of Gnutella

- Each peer has connection with C other peers
  - Approximate number of messages

    $$\sim \sum_{i=0}^{TTL-1} C(C-1)^i$$

    ~ 10^4 for values of $C=4$ and $TTL= 7$

    ~ O(N) if TTL = diameter of the graph

- Improving Gnutella:
  - Expanding ring
    - Start search with small TTL (e.g., 1)
    - If no success then increase TTL
  - k Random walkers
    - Forward query to a random neighbor
    - Start k random walkers with large TTL
    - Random walkers occassionally check if to continue
      - Avoids most of the duplicate messages of flooding
      - Has much higher latency

Gnutella - Recap

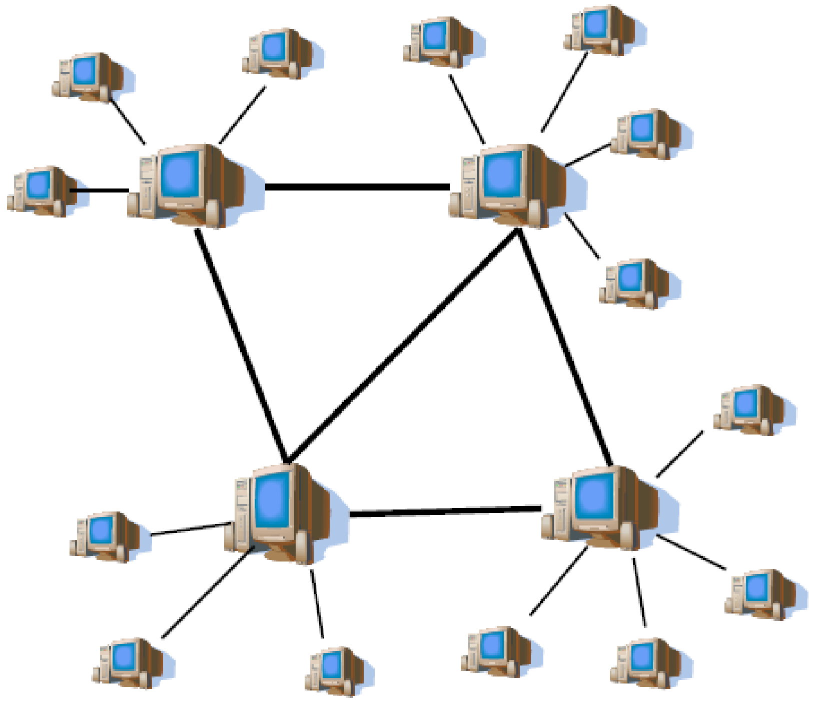
- Advantages/Disadvantages:
  - Advantages
    - Robustness (Almost impossible to destroy)
    - Low routing delay
      - (expected diameter of such overlays is usually in the order of log(N), where N is the number of nodes)
  - Disadvantages
    - Worst case O(N) message per lookup
    - No guarantees to find data item (due to TTL)

Super-Peer Networks

Super Peer Networks (1)

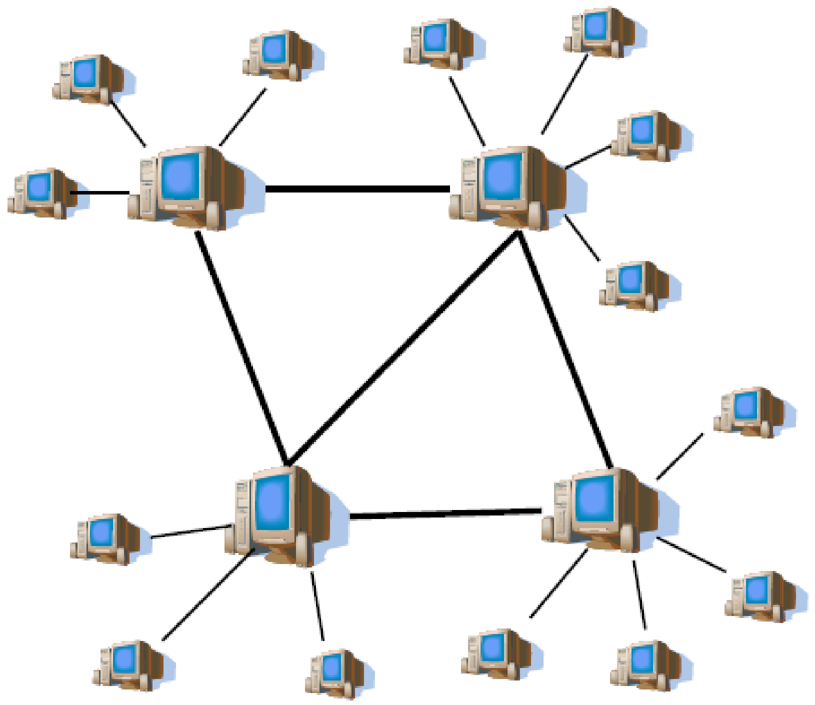
All animals are equal, but some animals are more equal than others.

– George Orwell (Animal Farm)



- Observations
  - Heterogeneity in the system
    - So why not exploit it?
  - Hierarchical system
  - Best of the two worlds
    - Napster & Gnutella

Super Peer Networks (2)

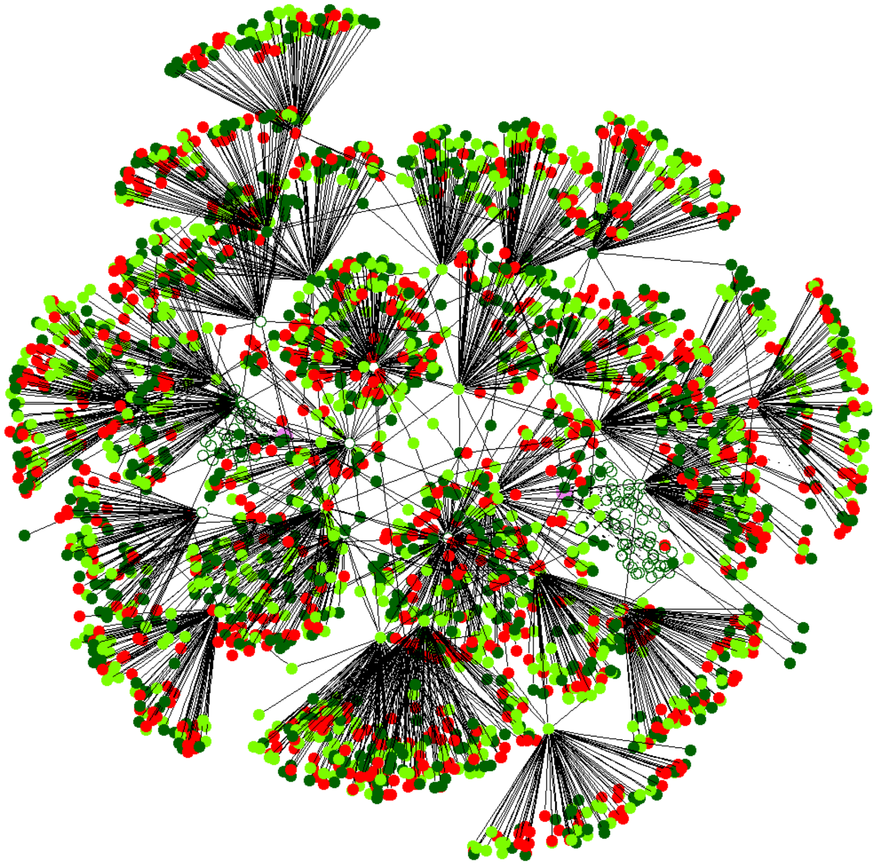


How does it work?

- Multiple index servers (super-peers)
  - Clients associated with one (or more) superpeers
    - Similar to Napster model
      - But multiple super peers per client is good for fault tolerance
  - Super-peers communicate with each other by message flooding
    - Similar to Gnutella model
  - Kazaa, Skype

Super Peer Networks (3)

- How does it work?
  - Low search latency
  - Recall
    - Good, but not guaranteed
    - No guarantee for consistency
  - Low(er) bandwidth consumption
    - Relatively small number of super-peers, so flooding overheads are much smaller
  - Storage cost
    - High at super-peers, low at clients
  - Resilience to failures
    - Moderate
    - Super-peers are easy target for attacks

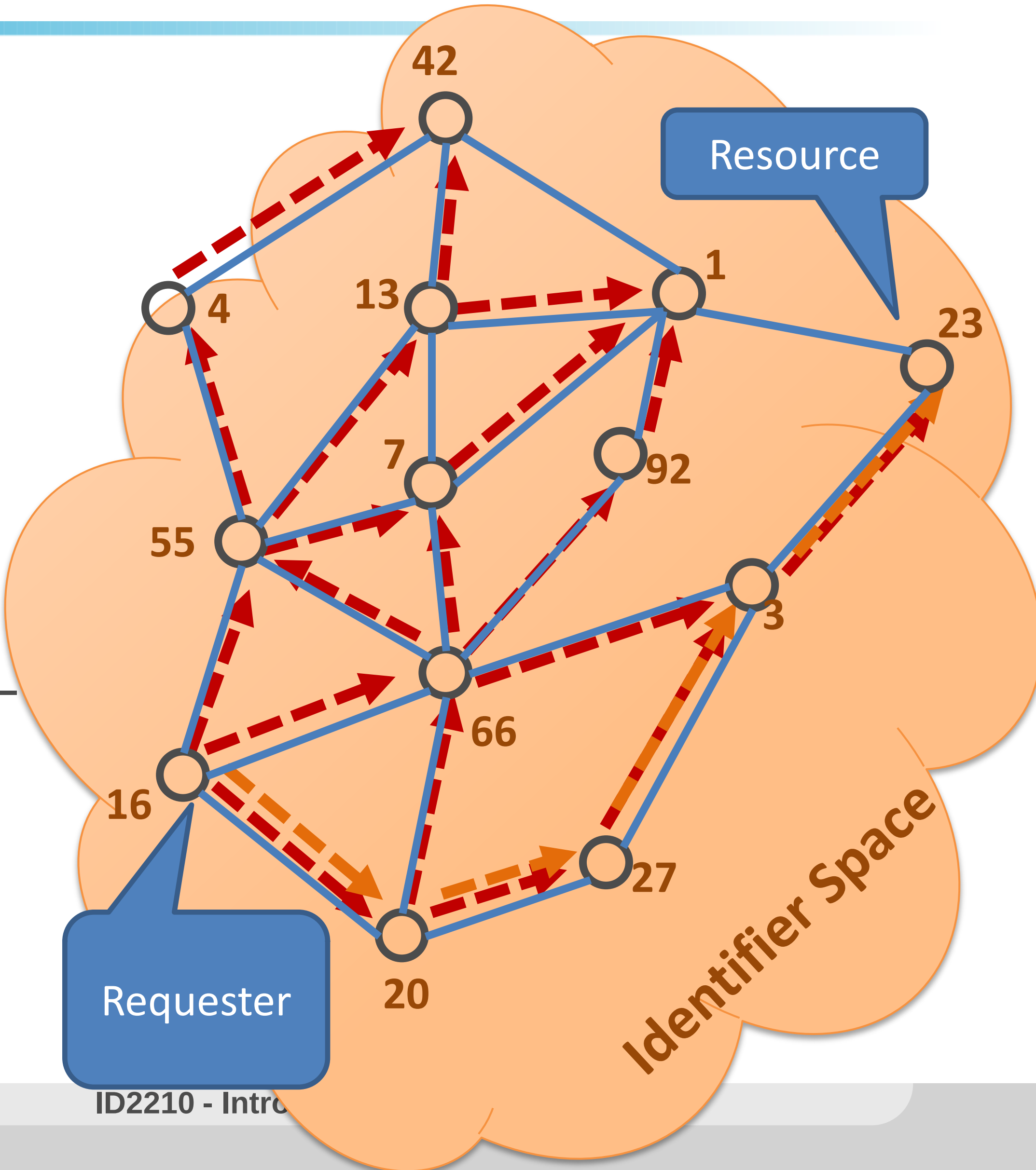


Structured Overlay Networks

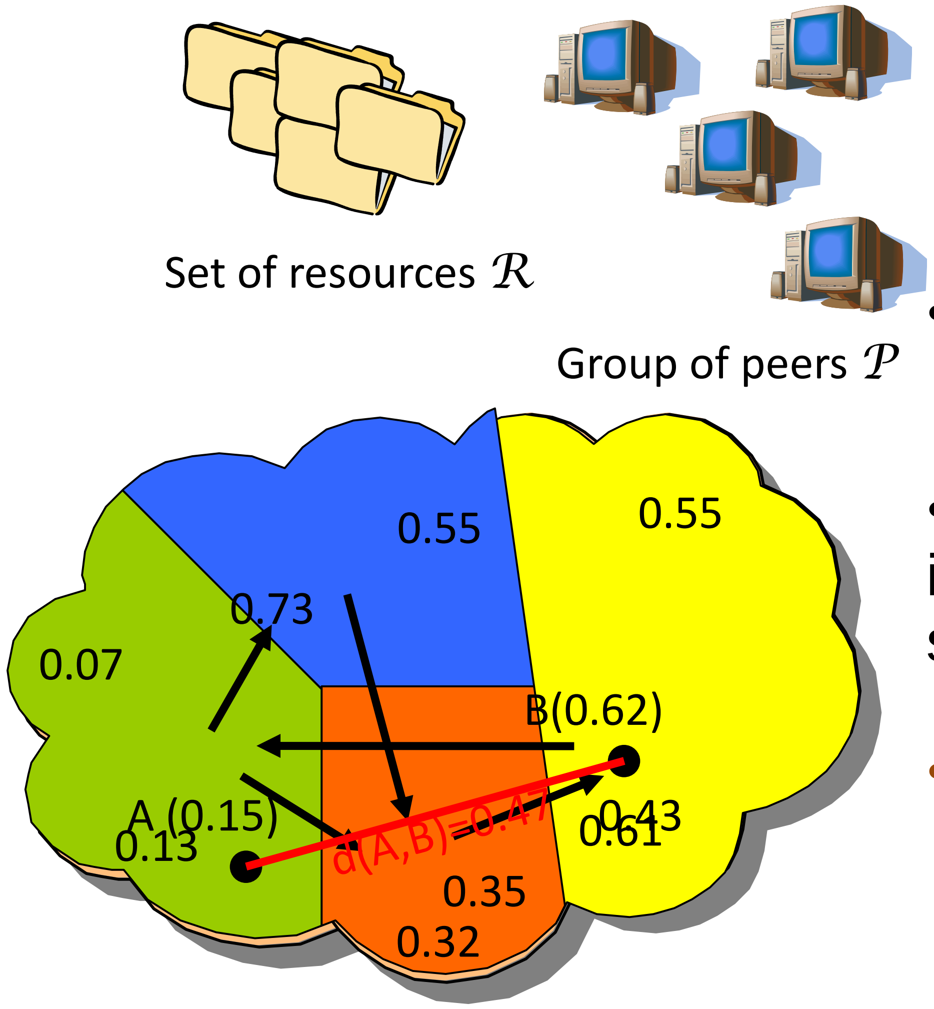
Structured Overlay Networks: Basic Principles

- Unstructured vs Structured Overlays

- Structured Overlays:
    - Identifier space embedded in a graph
    - Allows efficient search – graph becomes "navigable"



Conceptual Model of Structured P2P Overlays



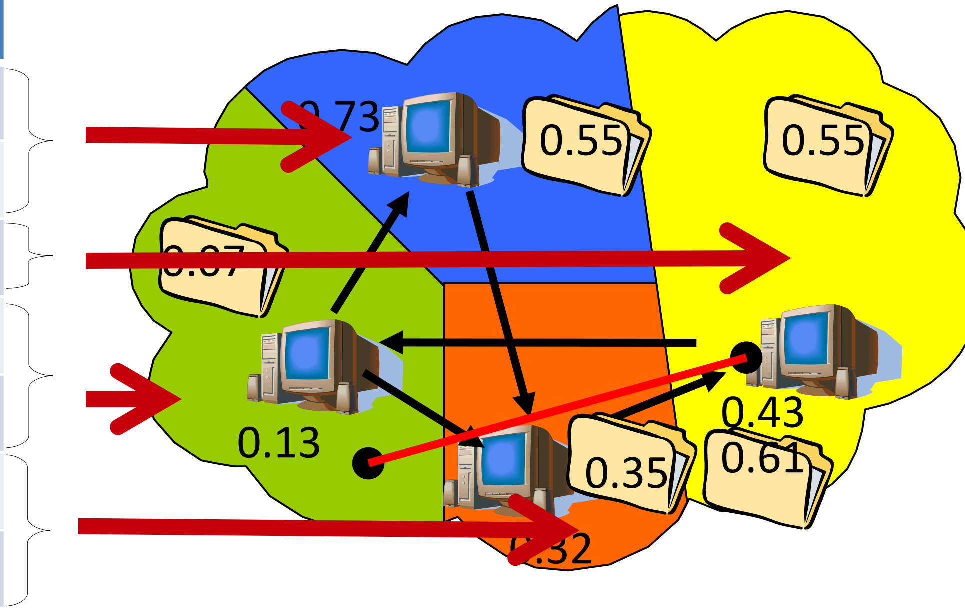
- Building steps:
  - Decide on common key space for nodes and resources
- Connect the nodes smartly
  - Greedy routing is possible
- Make a strategy for assigning items to nodes (dividing key space)
- **Most famous example: DHTs**

Distributed Hash Tables (DHT)

- An ordinary hashtable, which is...

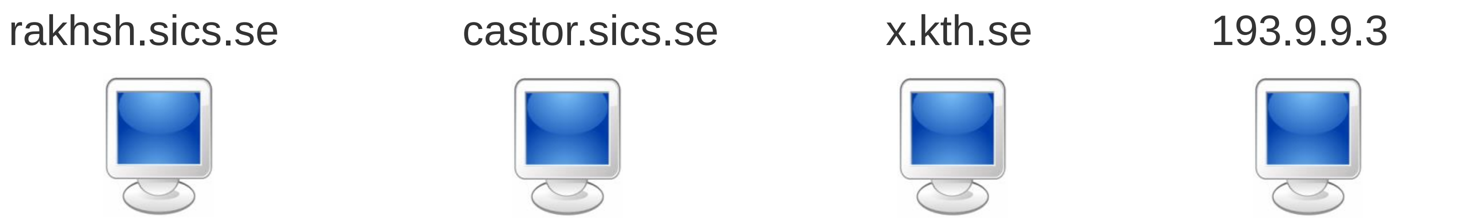
distributed.

Key	Value
Fatemeh	Stockholm
Amir	Tehran
Tallat	Islamabad
Cosmin	Bucharest
Seif	Stockholm
Sarunas	Vilnius
Jim	Dublin

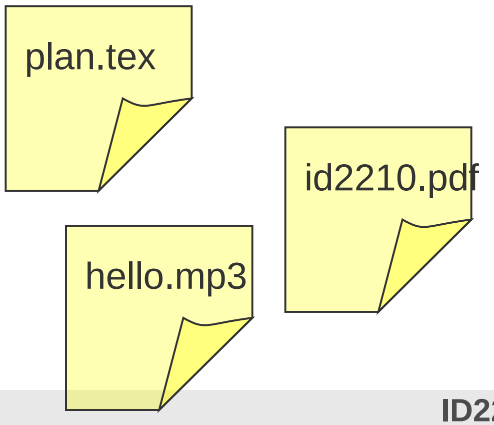


Assigning identifiers to nodes and resources

- Identifier space
  - E.g. of size 16, [0, 15].
  - Or a unit ring [0, 1)
- Uniform hashing: e.g, SHA-1 function



H(rakhsh.sics.se)=12 H(castor.sics.se)=3 H(x.kth.se)=0 H(192.9.9.3)=7



H(plan.tex)=2

H(id2210.pdf)=12

H(hello.mp3)=14

Distributed Hash Tables (DHT)

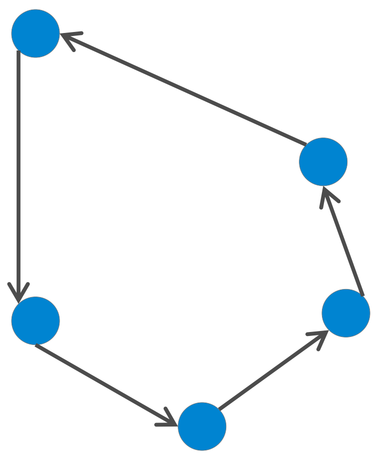
- A simple API:
  put(key,value)
  get(key)

- The neighbours of a node are well-defined (not randomly chosen).

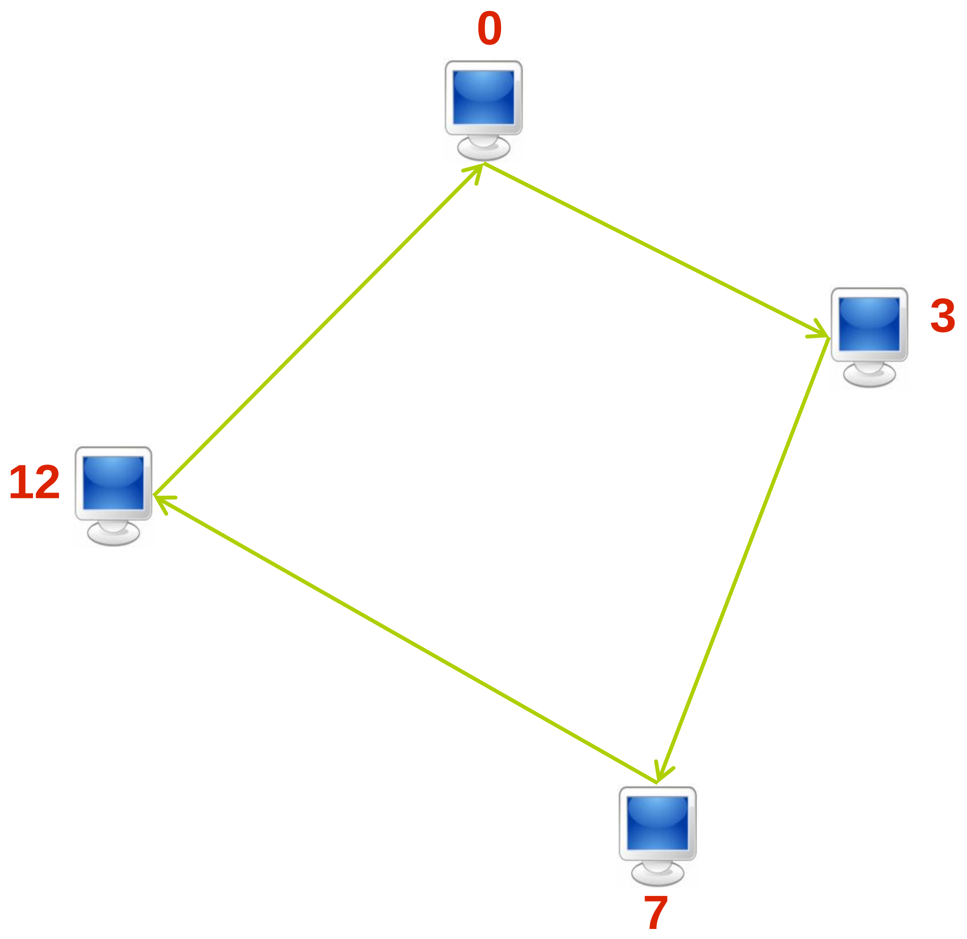
- Values are no longer stored at their owners, instead the network chooses at which node a data item will be stored

- a lookup operation can be performed from any node

- Nodes keep routing pointers
  - If item not found, route to another node

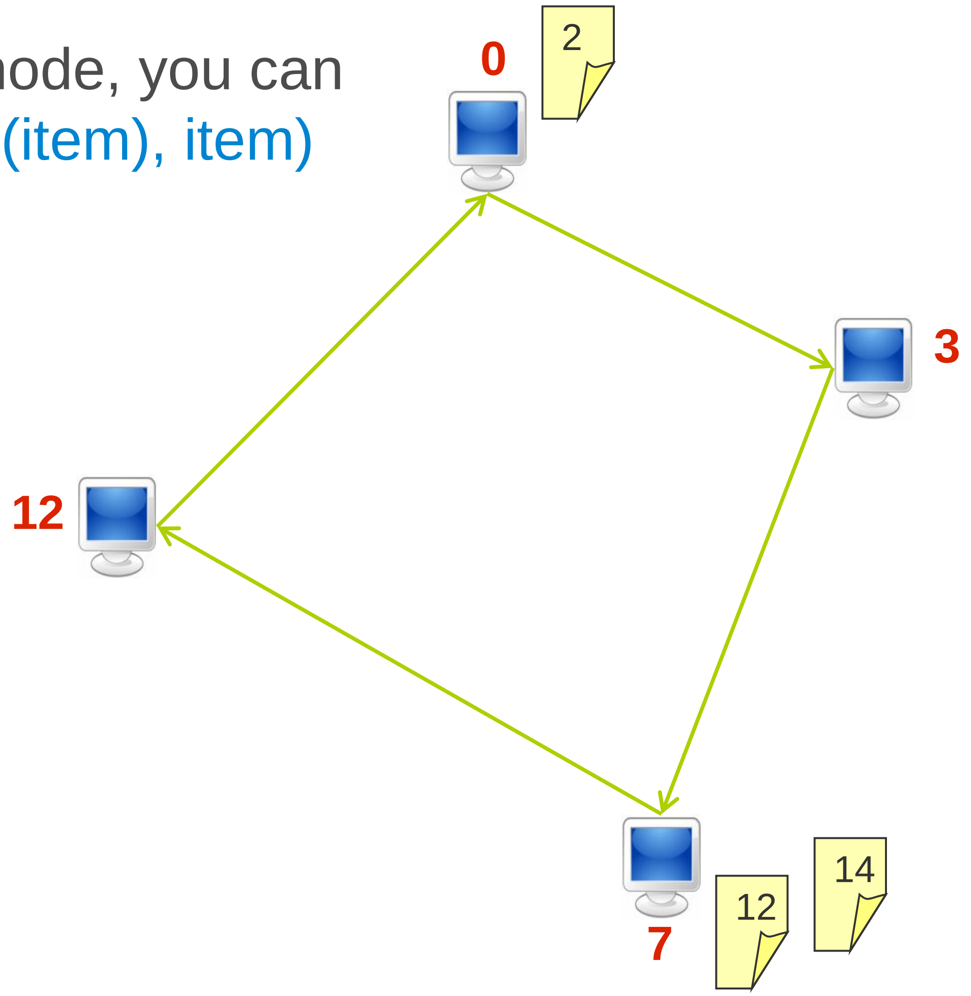


Connecting the nodes together (3/6)



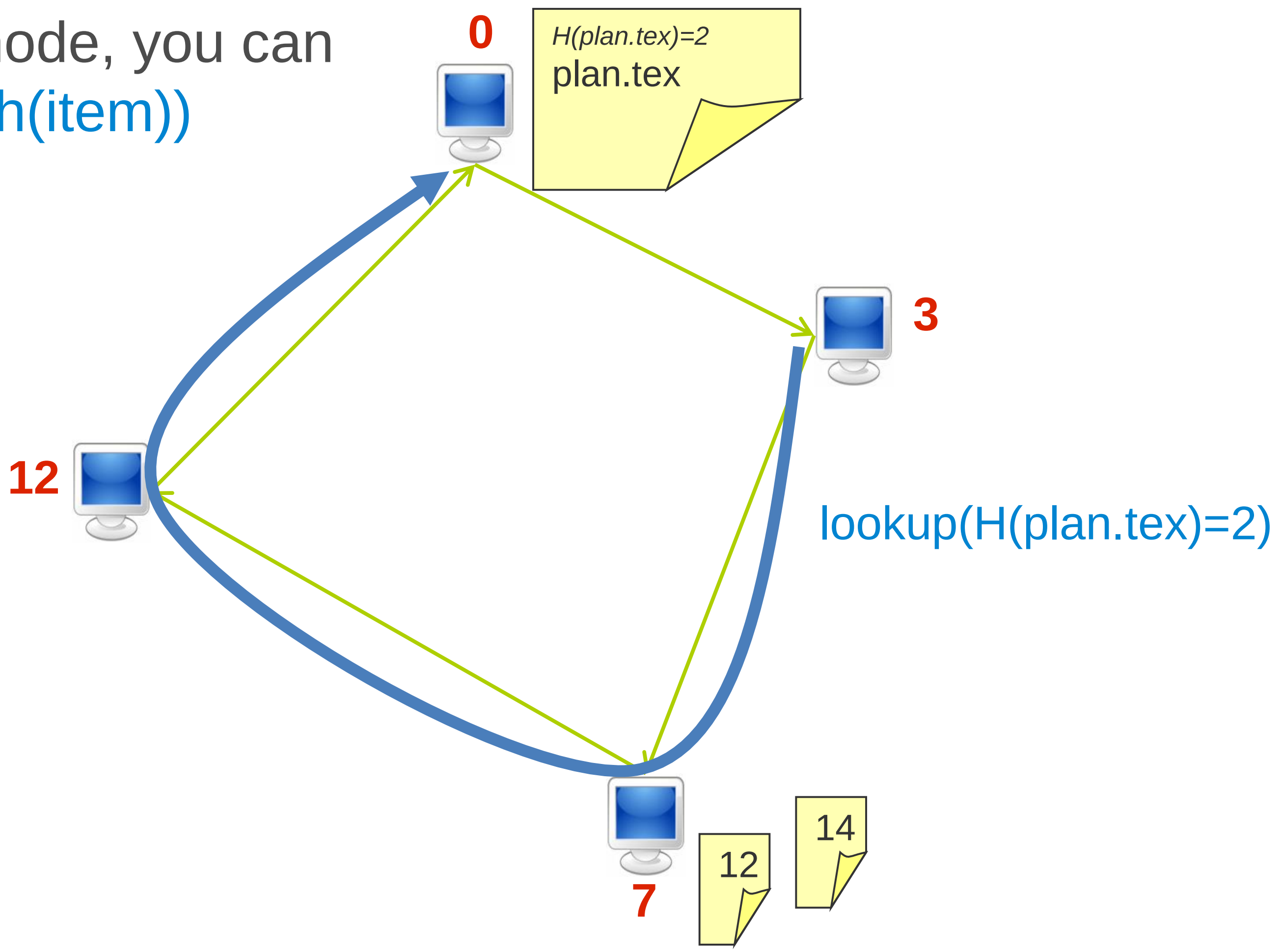
Assigning items to nodes (4/6)

- From any node, you can
- insert(hash(item), item)

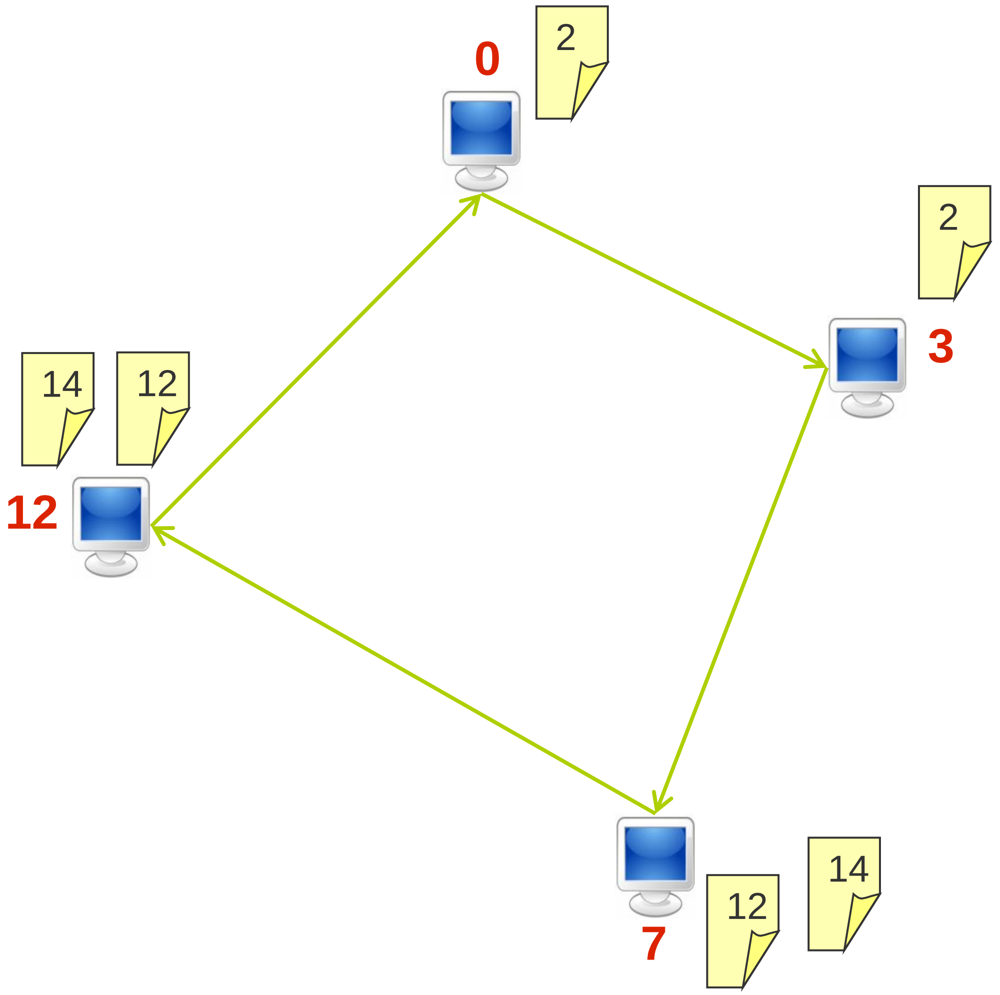


Looking up for data (5/6)

- From any node, you can
- lookup(hash(item))

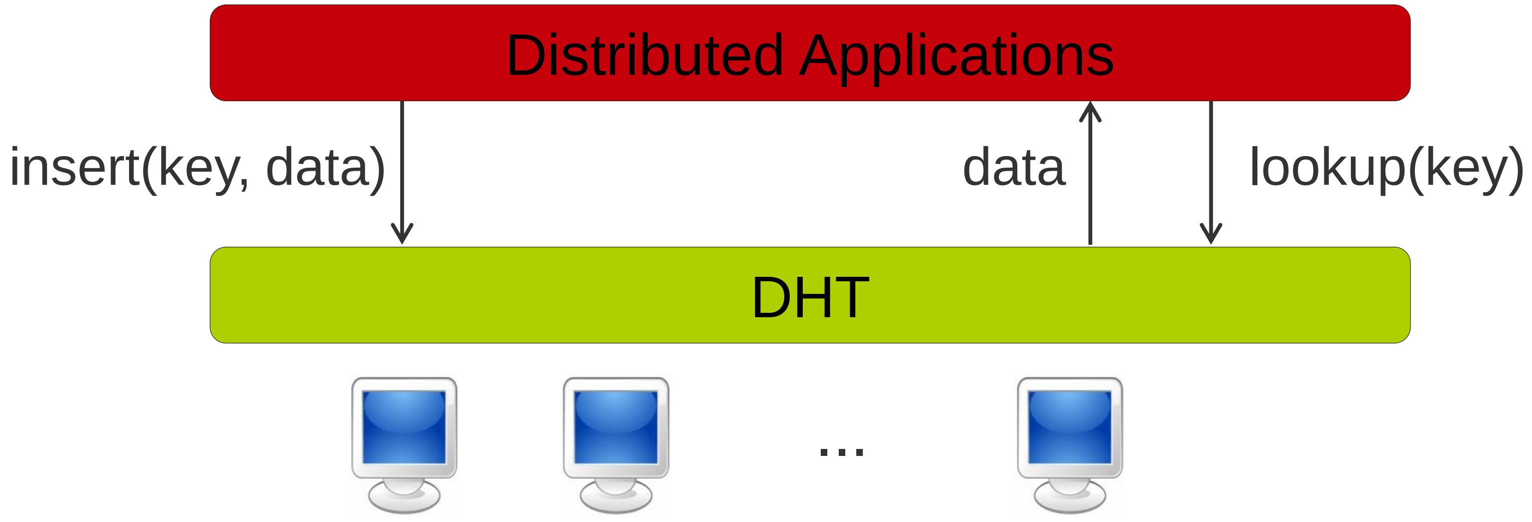


Replicate (6/6)



Distributed Hash Tables (DHT)

- Nodes are the hash buckets. See TreeMap.java from the JDK.
- Key identifies data uniquely
- DHT balances keys and data across nodes (because of uniform hashing)
- DHT replicates, caches, routes lookups, etc.
- Minimal interface
    - insert(key, data)
    - lookup(key)



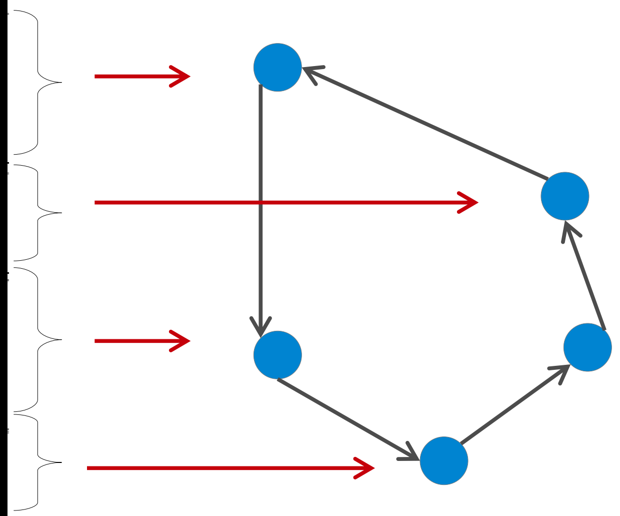
DHT Applications

- DHTs support a wide range of applications, because
  - Keys have (usually) no semantic meaning
  - Values are application dependent
- Examples:
  - Distributed File Systems [CFS, OceanStore, PAST, Arla/DKS]
  - Web cache/archives [Squirrel]
  - Event notification [Scribe, DKS]
  - Naming systems [ChordDNS, INS]
  - Query and indexing [Kademlia]
  - Backup store [HiveNet]
  - Distributed Authorizations Delegation

A Sample Application: Distributed Backup

- Clients install the backup tool
- Decide on amount of space to share
- Choose files for backup
- Data is encrypted
- Stored in the directory

Key	Value
Hi.mp3	2343
2210.txt	2511
Bye.avi	4539
...	...
...	...
...	...



DHT Advantages/ Disadvantages

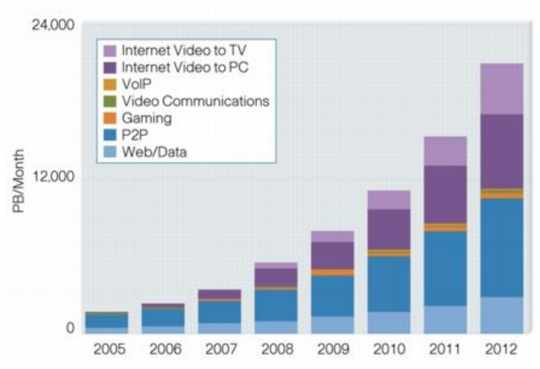
- DHT Advantages/ Disadvantages: ?

DHT Advantages/ Disadvantages

- DHT Advantages/ Disadvantages:
  - Advantages:
    - Supports large scale workloads
    - Efficient routing
    - Low-cost deployment
      - Self-organizing across administrative domains
      - Allows to be shared among applications
  - Disadvantages:
    - Maintenance cost

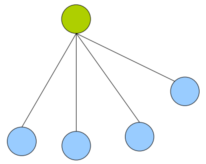
Summary

- P2P computing
  - Resource Sharing
  - Dual client/server role
  - Decentralization/Autonomy
  - Scalability
  - Robustness/Self-Organization

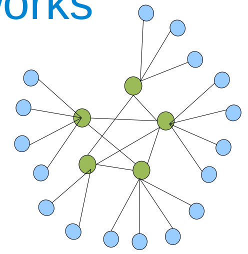


- Classes of P2P Systems:

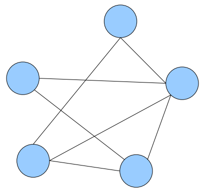
Centralized P2P



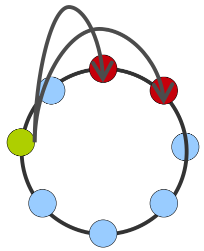
Super peer Networks



Unstructured P2P



Structured P2P



Questions?

Acknowledgements:

Some slides were derived from the lecture notes of K. Aberer (EPFL, Switzerland) and A. Datta (NTU, Singapore)