Freie Universität Berlin

Peer-to-peer systems

Netzprogrammierung

(Algorithmen und Programmierung V)

Where are we on our topic map?

Descriptive models for distributed system design

- Physical model
- Architectural model
  - Architectural elements
    - Communicating entities
      - Processes
      - Objects
      - Components
      - Web Services
    - Communication paradigm
      - Inter-process communication
        - UDP sockets
        - TCP sockets
        - Multi-cast
      - Indirect communication
      - Remote invocation
    - Roles and responsibilities
      - **Architectural styles**
        - **Client-server**
        - **Peer-to-peer**
    - Placement
      - Multiple server
      - Proxy/Cache
      - Mobile code
  - Architectural patterns
    - Vertical distribution
      - Multi-tier
      - Thin/Fat Client
    - Horizontal distribution
- Interaction model
  - Interaction model
  - Failure model
  - Security model

Agenda

Basics of peer-to-peer systems: motivation, characteristics, and examples

Distributed object location and routing in peer-to-peer systems

Unstructured peer-to-peer systems

- Napster
- Gnutella

Structured Peer-to-Peer systems based on the concept of distributed hash tables

- Pastry

Peer-to-peer systems

Introduction

Motivation

Peer-to-peer systems (P2P systems) represent a **paradigm** for the construction of distributed systems and applications in which data and computational resources are contributed by many hosts on the Internet.

P2P systems enable the **sharing of data and resources on a very large scale** by eliminating any requirement for separately managed servers and their associated infrastructure.

P2P systems have been used to provide file sharing, web caching, information distribution and other services, exploiting the resources of tens of thousands of machines across the Internet.

Freie Universität Berlin

Characteristics of peer-to-peer systems

The design of P2P systems ensure that each **user contributes resources** to the system.

Although user may differ in the resources that they contribute, all the nodes in a P2P system have the **same functional capability and responsibility**.

The correct operation of a P2P system does **<u>not</u>** depend on the existence of any centrally administered systems.

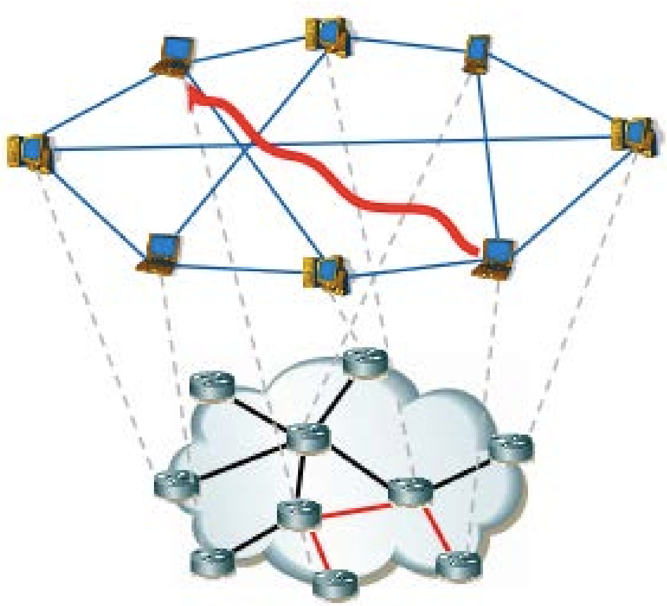
A key issue for the efficient operation of an P2P system is the choice of the **algorithm for the placement of data across many hosts** and **subsequent access** to it in a manner that **balances the workload** and **ensures availability** <u>without adding undue overheads</u>.

Distributed object location and routing

The operation of any peer-to-peer content distribution system relies on a network of peer computers (nodes) and connections (edges) between them.

This network is formed on top of-and independently from-the underlying physical computer (typically IP) network, and is thus referred to as an **"overlay" network**. (also see lecture about publish subscribe systems).

The topology, structure and degree of centralization of the overlay network, and the routing and location mechanisms it employs for messages and content are crucial to the operation of the system.



Overlay networks can be distinguished in terms of their

- **Centralization** and
- **Structure**

Overlay network centralization

Purely decentralized architectures

- All nodes in the network perform exactly the same tasks, acting both as servers and clients ("servents")
- There is no central coordination of their activities

Hybrid decentralized architectures

- Some of the node are supernodes, acting as local central indexes
- Supernodes are dynamically assigned (varies between different systems) and if they fail they are automatically replaced with others

Centralized architectures

- Central server facilitating the interaction between peers
- Server maintains directories of meta-data describing the shared files stored by the peer nodes
- Server performs the lookups and identifying the nodes storing the files

Unstructured overlay network

The placement of content (files) is completely unrelated to the overlay topology.

In an unstructured network, content typically needs to be located.

- Location of resource only known to submitter
- Peers & resources have no special identifier
- Each peer is responsible only for the resources it submitted
- Introduction of new resource at any location

The main task is to search

- Find all peers storing/being in charge of resources fitting to some criteria
- Direct communication when peers have been identified

Examples: Napster, Gnutella

Structured overlay network

The overlay topology is tightly controlled and files (or pointers to them) are placed at precisely specified locations. These systems essentially provide a mapping between content (e.g. file identifier) and location (e.g. node address), in the form of a **distributed routing table**.

- Location of resources not only known to submitter
- Each peer may well be responsible for resources it has not submitted
- Introduction of new resource(s) at specific location, i.e. to give peers and resources (unique) identifiers
- PeerIDs and ObjectIDs (RessourceIDs) should be from the same key set (**globally unique identifiers GUIDs**)
- Each peer is responsible for a specific range of ObjectIDs (i.e., RessourceIDs)

The main task is to lookup

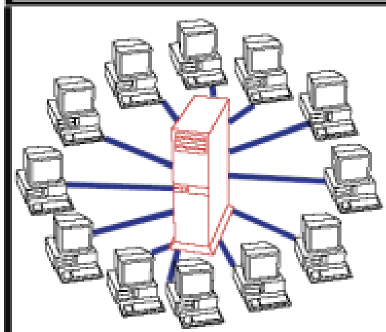
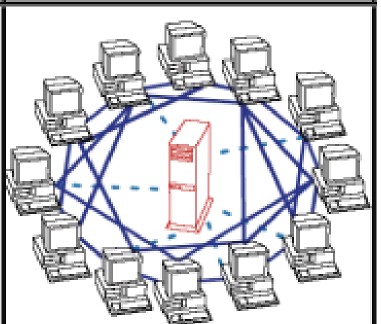
- To “route” queries across the overlay network to peers with specific IDs

Example: Pastry

<table>
<tr><th>Client-Server</th><th colspan="3">Peer-to-Peer</th></tr>
<tr><td rowspan="3"></td><td colspan="3">1. Resources are shared between the peers
2. Resources can be accessed directly from other peers
3. Peer is provider and requestor (Servent concept)</td></tr>
<tr><th colspan="2">Unstructured P2P</th><th>Structured P2P</th></tr>
<tr><th>1st Generation</th><th>2nd Generation</th><th></th></tr>
<tr><td>1. Server is the central entity and only provider of service and content.
→ Network managed by the Server
2. Server as the higher performance system.
3. Clients as the lower performance system

Example: WWW</td><td>Centralized P2P

1. All features of Peer-to-Peer included
2. Central entity is necessary to provide the service
3. Central entity is some kind of index/group database

Example: Napster</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

(Eberspächer, & Schollmeier 2005)

Peer-to-peer systems

Napster

Brief introduction into Napster

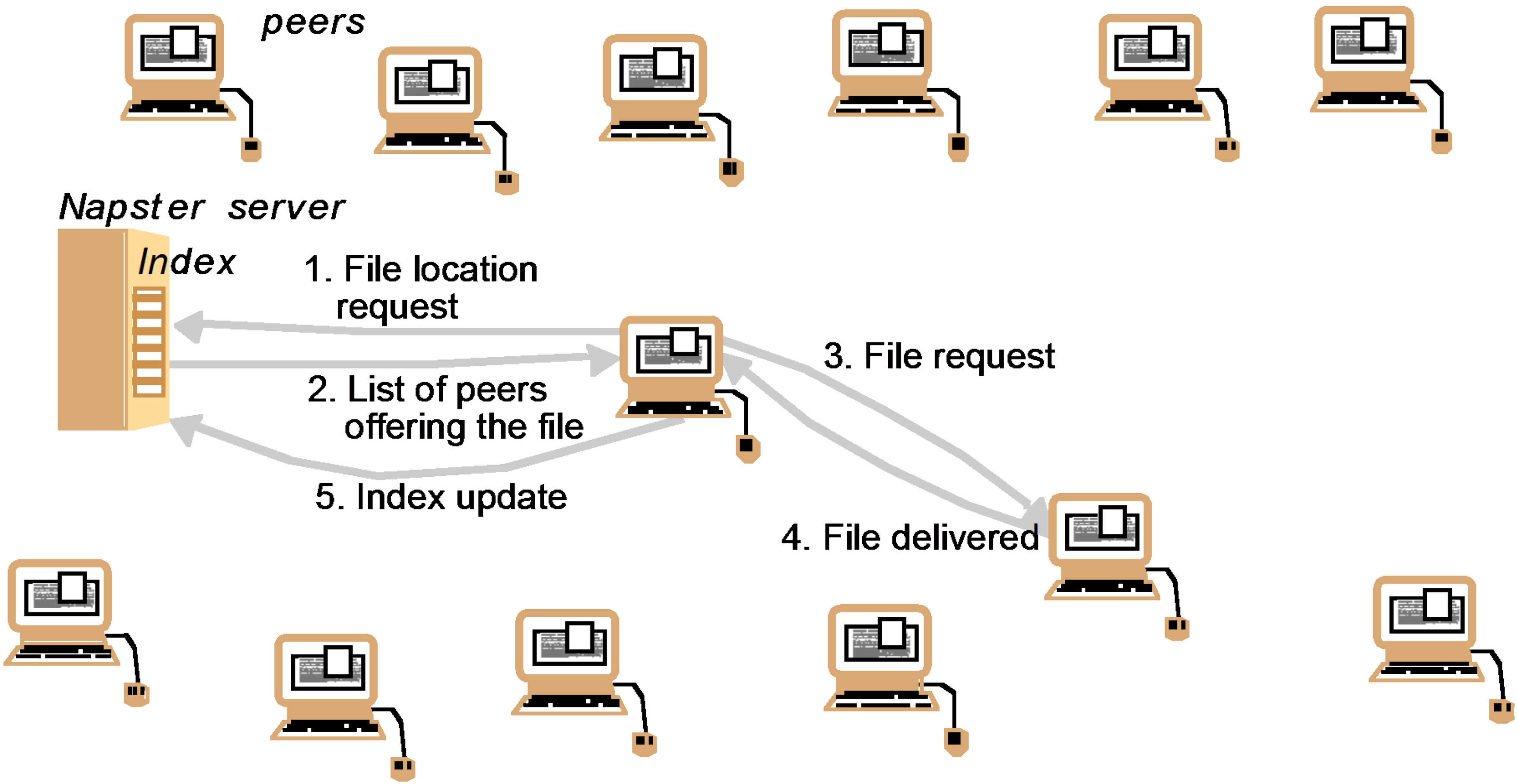
In June 1999, the first peer-to-peer file sharing system, Napster was released.

It is a **centralized unstructured peer-to-peer system** that requires a central server for indexing and peer discovery.

Napster provided a service where they indexed and stored file information that users of Napster made available on their computers for others to download, and the files were transferred directly between the host and client users after authorization by Napster.

July 2001 Napster was shut down as a result of legal proceedings.

Napster's method of operation



Lessons learned from Napster

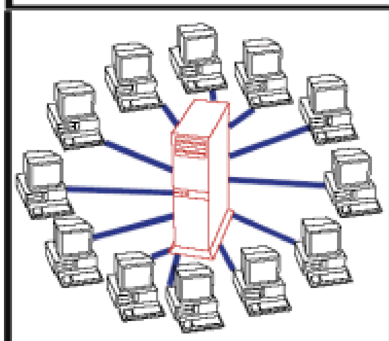
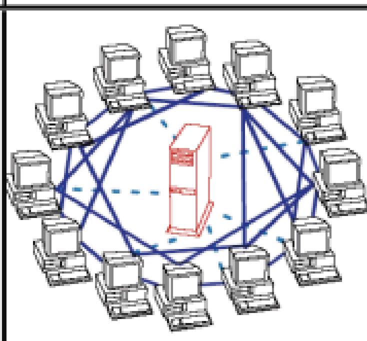
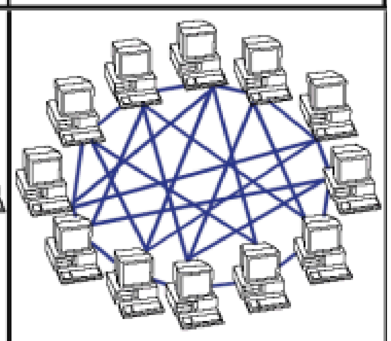
Napster took advantage of special characteristics of the application, such as music files are never updated, and no guarantees are required concerning the availability of individual files.

The advantage of centralized systems is that they are simple to implement and they locate files quickly and efficiently.

Their main disadvantage is that they are vulnerable to censorship, legal action, surveillance, malicious attack, and technical failure, since the content shared, or at least descriptions of it and the ability to access it are controlled by the single institution, company or user maintaining the central server.

Furthermore, these systems are considered **inherently unscalable**, as there are bound to be limitations to the size of the server database and its capacity to respond to queries. (Large web search engines have however repeatedly provided counterexamples to this notion.)

<table>
<tr><th>Client-Server</th><th colspan="4">Peer-to-Peer</th></tr>
<tr><td rowspan="3"></td><td colspan="4">1. Resources are shared between the peers
2. Resources can be accessed directly from other peers
3. Peer is provider and requestor (Servent concept)</td></tr>
<tr><th colspan="3">Unstructured P2P</th><th>Structured P2P</th></tr>
<tr><th colspan="2">1st Generation</th><th>2nd Generation</th><th></th></tr>
<tr><td rowspan="2">1. Server is the central entity and only provider of service and content.
→ Network managed by the Server
2. Server as the higher performance system.
3. Clients as the lower performance system

Example: WWW</td><th>Centralized P2P</th><th>Pure P2P</th><td rowspan="3"></td><td rowspan="3"></td></tr>
<tr><td>1. All features of Peer-to-Peer included
2. Central entity is necessary to provide the service
3. Central entity is some kind of index/group database
Example: Napster</td><td>1. All features of Peer-to-Peer included
2. Any terminal entity can be removed without loss of functionality
3. → No central entities
Examples: Gnutella 0.4, Freenet</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

(Eberspächer, & Schollmeier 2005)

Peer-to-peer systems

Gnutella 0.4

Introducing Gnutella

Gnutella is originally created by Justin Frankel of Nullsoft. As a **unstructured approach**, there is no overall control over the topology or the placement of objects within the network. Additionally, there is **no central coordination of the activities** in the network. Users connect to each other directly in a ad-hoc fashion through a software application.

Similarities between Gnutella and Napster

- Users place the files they want to share on their hard disks and make them available to everyone else for downloading in peer-to-peer fashion.
- Users run a piece of Gnutella software to connect to the Gnutella network.

Differences between Gnutella and Napster

- There is no central database that knows all of the files available on the Gnutella network. Instead, all of the machines on the network tell each other about available files using a distributed query approach.
- There are many different client applications available to access the Gnutella network.

Gnutella protocol messages

Broadcast Messages

- **Ping**: initiating message (“I’m here”)
- **Query**: search pattern and TTL (time-to-live)

Back-Propagated Messages

- **Pong**: reply to a ping, contains information about the peer
- **Query response**: contains information about the computer that has the needed file

membership

search

Node-to-Node Messages

- **GET**: return the requested file
- **PUSH**: push the file to me

file transfer

(Horowitz 2002)

Gnutella characteristics

Scalability

- When a node receives a ping/query message, it forwards it to the other nodes
- Existing mechanisms to reduce traffic

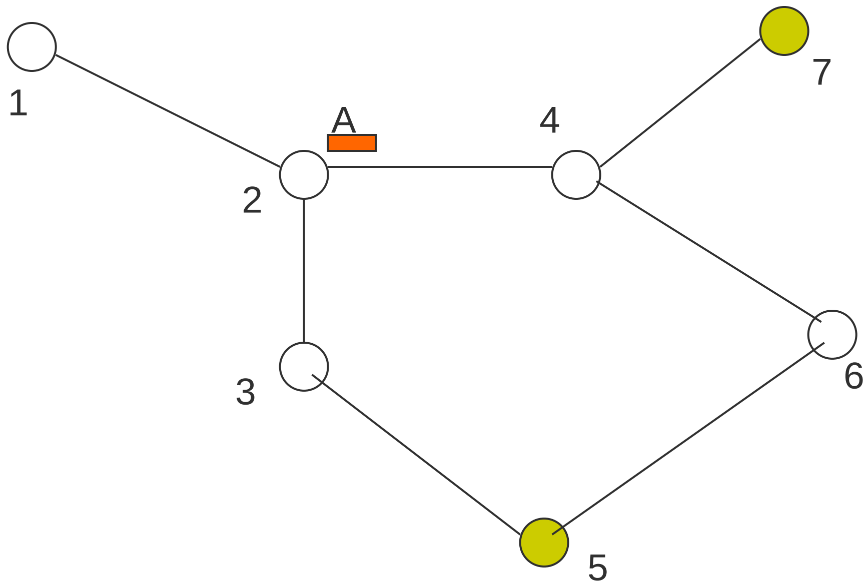
TTL counter

- Cache information about messages they received, so that they don't forward duplicated messages

Anonymity

- Gnutella provides for anonymity by masking the identity of the peer that generated a query

Gnutella search mechanism

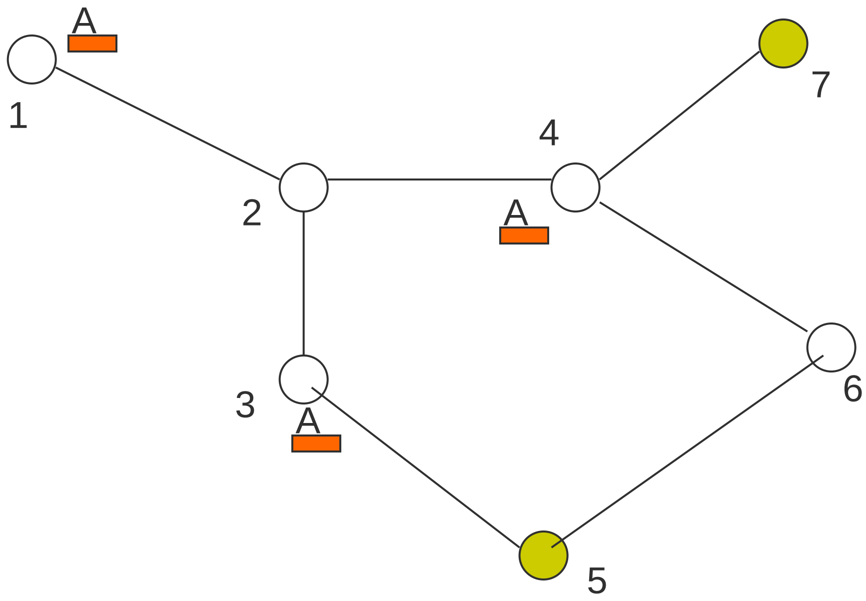


Steps:

1. Node 2 initiates search for file A

(Horowitz 2002)

Gnutella Search Mechanism

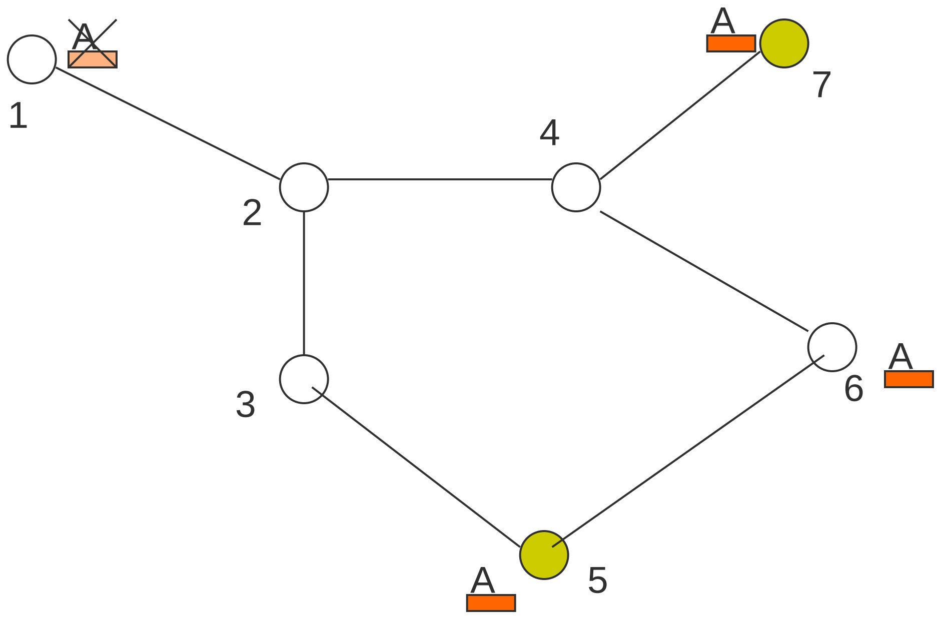


Steps:

1. Node 2 initiates search for file A
2. Sends message to all neighbors

(Horowitz 2002)

Gnutella Search Mechanism

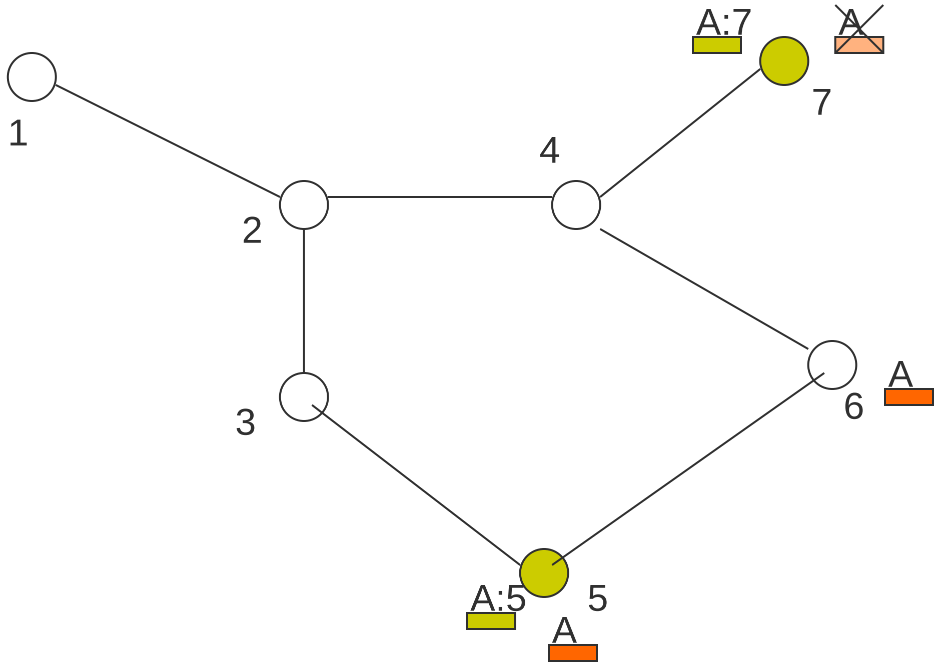


Steps:

1. Node 2 initiates search for file A
2. Sends message to all neighbors
3. Neighbors forward message

(Horowitz 2002)

Gnutella Search Mechanism

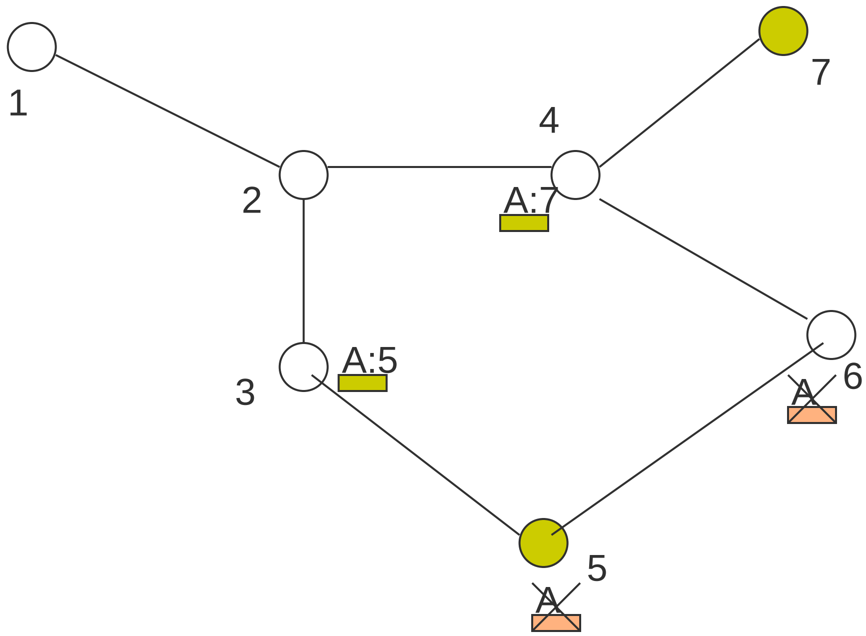


Steps:

1. Node 2 initiates search for file A
2. Sends message to all neighbors
3. Neighbors forward message
4. Nodes that have file A initiate a reply message

(Horowitz 2002)

Gnutella Search Mechanism

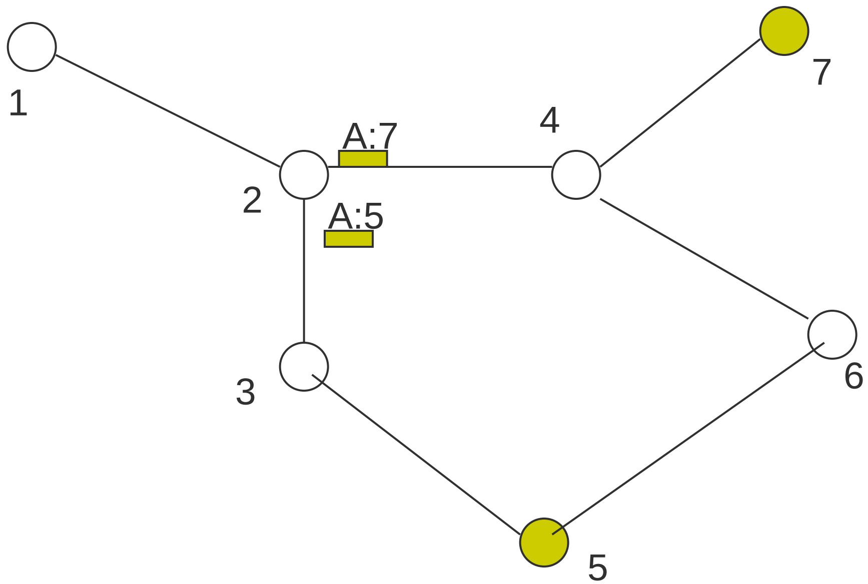


Steps:

1. Node 2 initiates search for file A
2. Sends message to all neighbors
3. Neighbors forward message
4. Nodes that have file A initiate a reply message
5. Query reply message is back-propagated

(Horowitz 2002)

Gnutella Search Mechanism

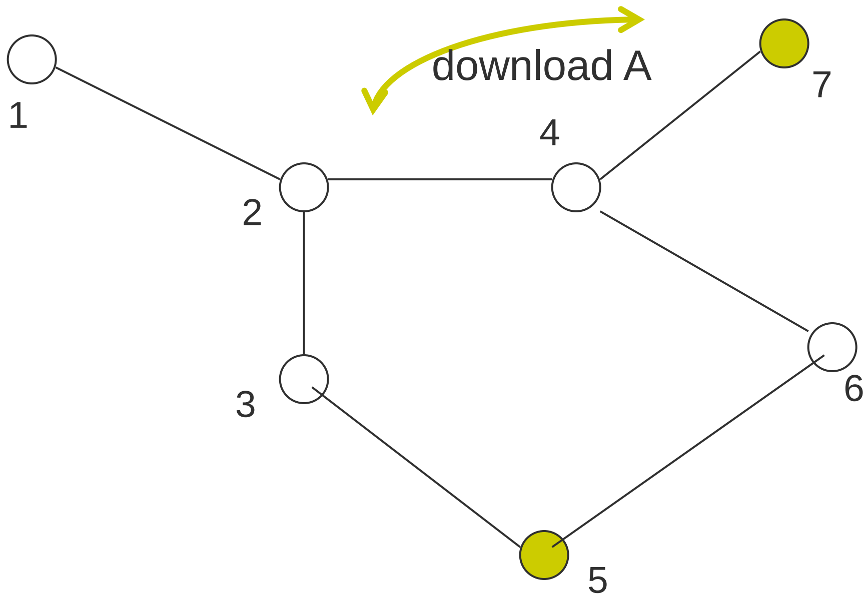


Steps:

1. Node 2 initiates search for file A
2. Sends message to all neighbors
3. Neighbors forward message
4. Nodes that have file A initiate a reply message
5. Query reply message is back-propagated

(Horowitz 2002)

Gnutella Search Mechanism



Steps:

1. Node 2 initiates search for file A
2. Sends message to all neighbors
3. Neighbors forward message
4. Nodes that have file A initiate a reply message
5. Query reply message is back-propagated
6. File download

(Horowitz 2002)

Gnutella search strategy: Flooding

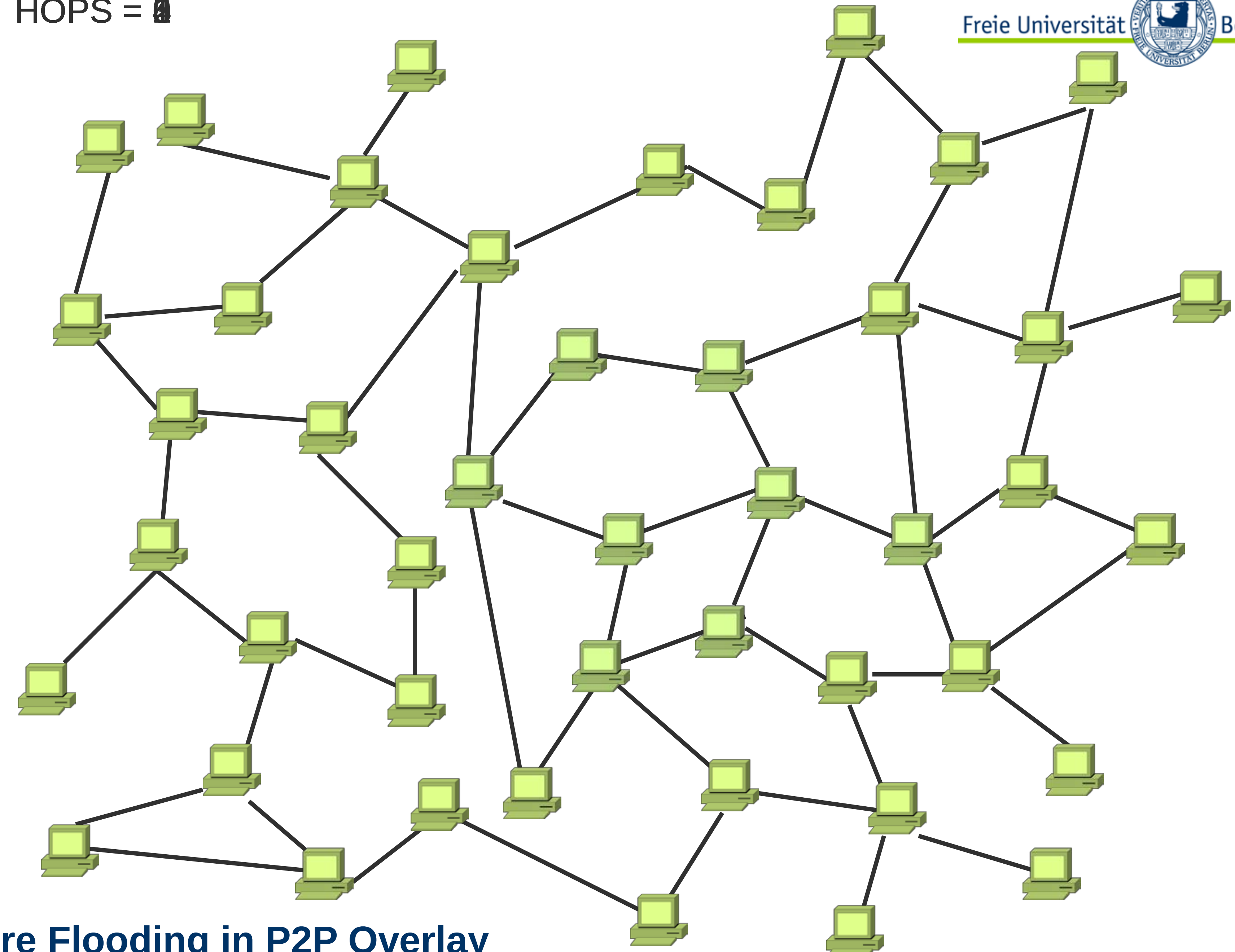
Simple and robust

- No state maintenance needed
- High tolerance to node failures

Effective and of low latency

- Always find the shortest / fastest routing paths

HOPS = [illegible]



Pure Flooding in P2P Overlay

Gnutella search strategy: Flooding

Simple and robust

- No state maintenance needed
- High tolerance to node failures

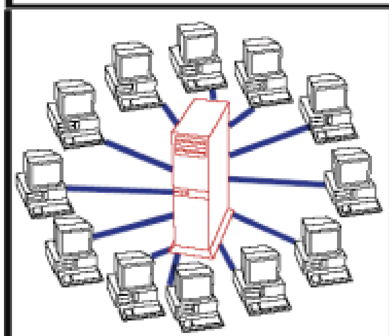
Effective and of low latency

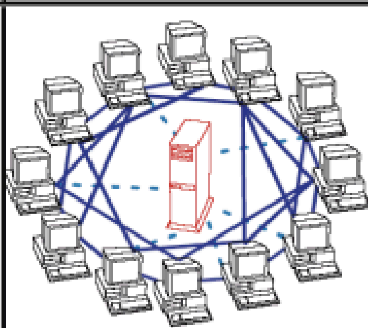
- Always find the shortest / fastest routing paths

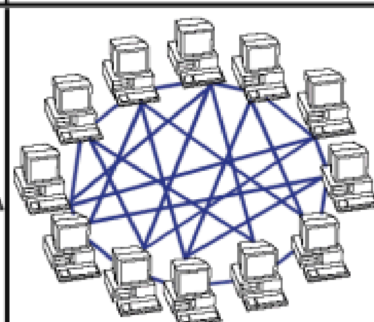
Problems of Flooding

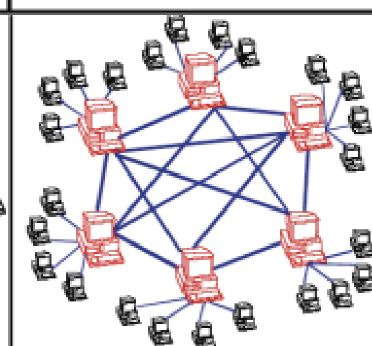
- Loops in Gnutella networks
    - Caused by redundant links
    - Result in endless message routing
- Current solutions by Gnutella
    - Detect and discard redundant messages
    - Limit TTL (time-to-live) of messages

Client-Server	Peer-to-Peer			
	1. Resources are shared between the peers 2. Resources can be accessed directly from other peers 3. Peer is provider and requestor (Servent concept)			
	Unstructured P2P			Structured P2P
	1st Generation		2nd Generation	
	Centralized P2P	*Pure P2P*	*Hybrid P2P*	
1. Server is the central entity and only provider of service and content. → Network managed by the Server 2. Server as the higher performance system. 3. Clients as the lower performance system Example: WWW	1. All features of Peer-to-Peer included 2. Central entity is necessary to provide the service 3. Central entity is some kind of index/group database Example: Napster	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities Examples: Gnutella 0.4, Freenet	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → dynamic central entities Example: Gnutella 0.6, JXTA	









(Eberspächer, & Schollmeier 2005)

Peer-to-peer systems
Gnutella 0.6

Improvements of the new protocol

The new protocol implements a **unstructured, hybrid architecture**.

All peers still cooperate to offer the service but some nodes, i.e. **ultrapeers**, are designated to have additional resources.

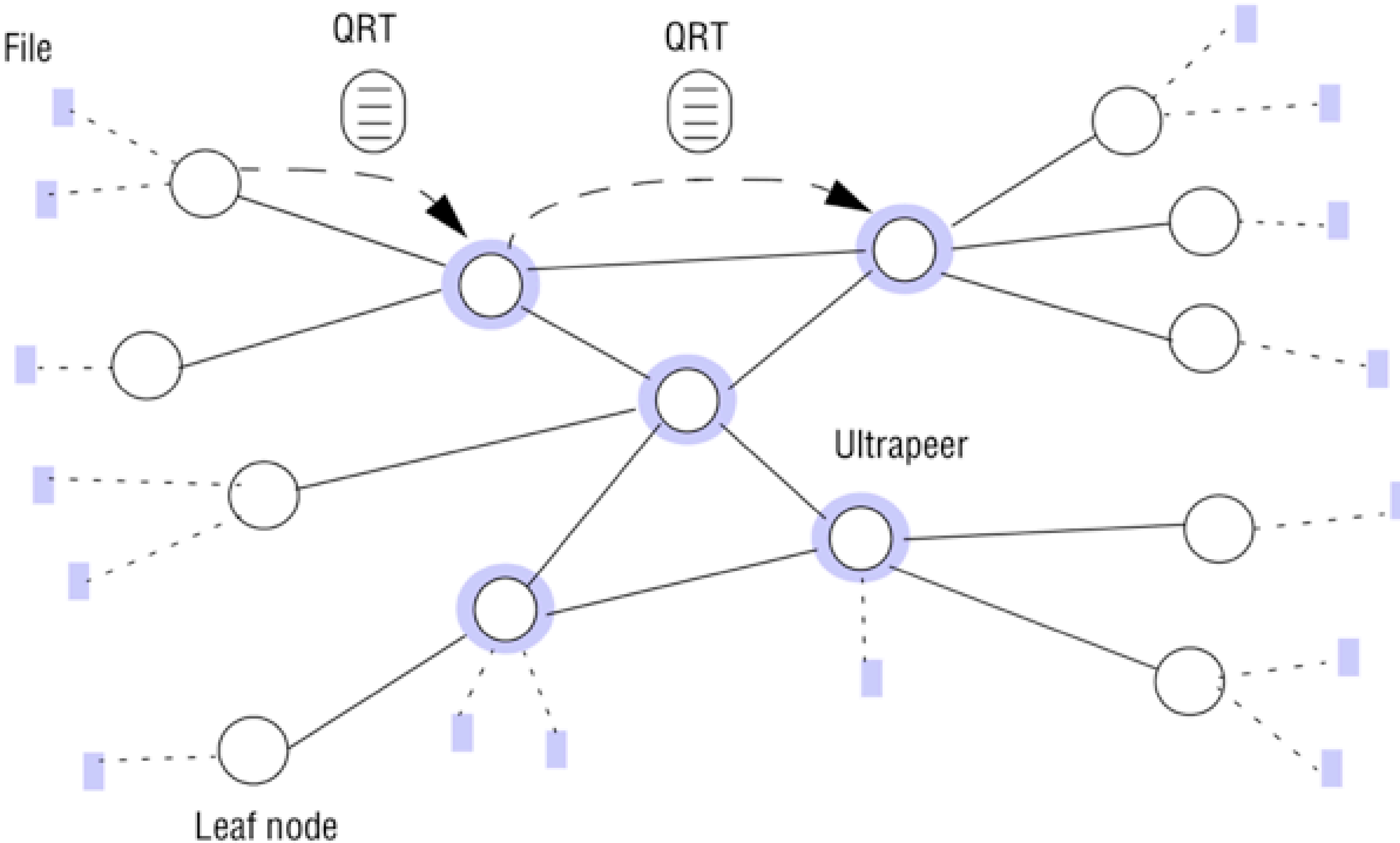
Normal nodes, i.e. leaves, connect themselves to a small number of ultrapeers which are heavily connected to other ultrapeers (> 32 connections).

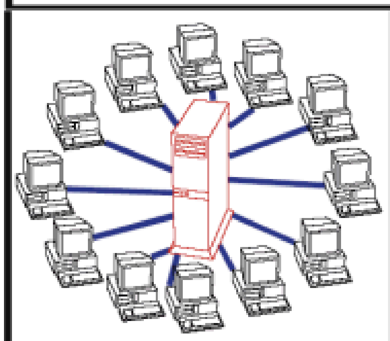
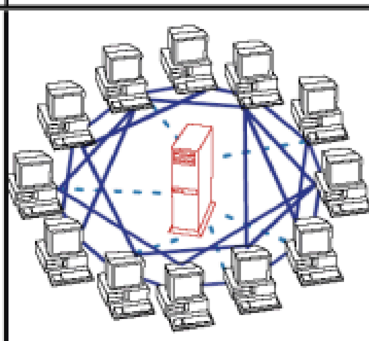
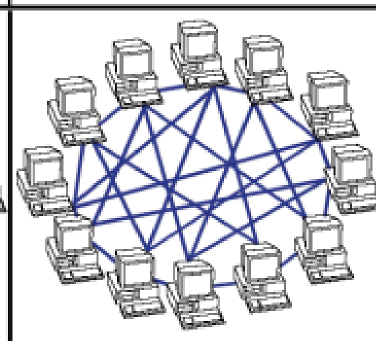
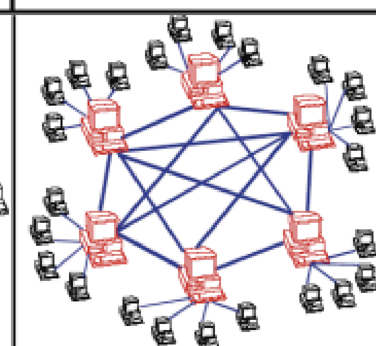
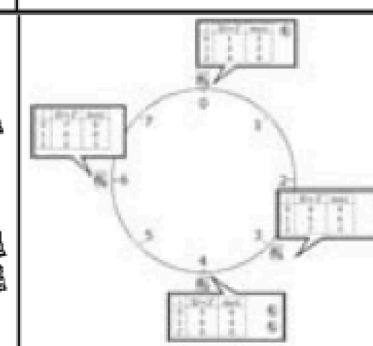
=> The maximal number of hops required for exhaustive search is dramatically reduced.

A new protocol has been introduced: the **Query Routing Protocol (QRP)** which has been designed to reduce the number of queries issued by each node.

Additionally, each node produces a **Query Routing Table (QRT)** containing the hash values representing the the files available on that node.

Key elements in the Gnutella 2 protocol



Client-Server	Peer-to-Peer			
	1. Resources are shared between the peers 2. Resources can be accessed directly from other peers 3. Peer is provider and requestor (Servent concept)			
	Unstructured P2P			**Structured P2P**
	1st Generation		**2nd Generation**	
1. Server is the central entity and only provider of service and content. → Network managed by the Server 2. Server as the higher performance system. 3. Clients as the lower performance system Example: WWW	*Centralized P2P*	*Pure P2P*	*Hybrid P2P*	*DHT-Based*
	1. All features of Peer-to-Peer included 2. Central entity is necessary to provide the service 3. Central entity is some kind of index/group database Example: Napster	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities Examples: Gnutella 0.4, Freenet	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → dynamic central entities Example: Gnutella 0.6, JXTA	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities 4. Connections in the overlay are "fixed" Examples: Chord, CAN
				

(Eberspächer, & Schollmeier 2005)

Peer-to-peer systems

Pastry

Overview about Pastry

P2P overlay that is **using Distributed Hash Tables (DHT)** with **prefix-based routing with both peer ID and object ID**.

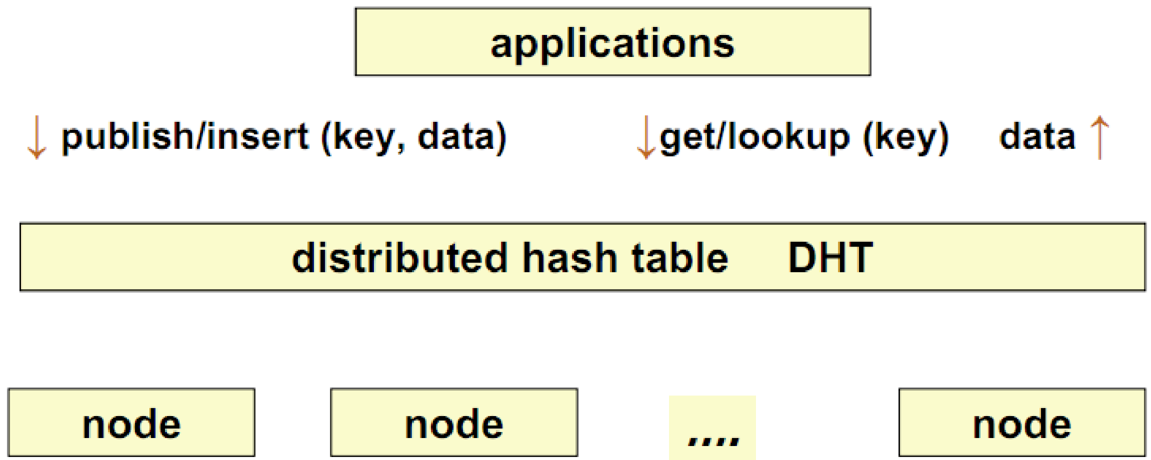
Prefix routing narrows the search for the next node along the route by applying a binary mask that selects an increasing number of hexadecimal digits from the destination GUID (Globally Unique ID – typically 128-bit secure hash) after each hop.

It is originally developed by Microsoft and Rice Uni, but a free version (FreePastry) exists that is a prototypical Implementation of Pastry. The latter is mostly used by scientific community.

Similar algorithms are Chord and CAN.

Mode of operation of a distributed hash table

every node stores and maintains part of hash table



lookup(key) -> node or data directly

Lookup where it is stored and how it is identified there

every object / resource has a (hash) key which is stored at node responsible for its key

Distributed hash table: steps of operation

1. Mapping of nodes and data in the same address space
   - Peers and content are addressed using flat identifiers (GUIDs)
   - Common address space for data and nodes
   - Nodes are responsible for data in certain parts of the address space
   - Note: Association of data to nodes may change since nodes may disappear

2. Storing / Looking up data in the DHT
   - "Look-up" for data = routing to the responsible node
   - Note: Responsible node not necessarily known in advance
   - Deterministic statement about availability of data

Example - First four rows of a Pastry routing table

Routing table at a node whose GUID begins 65A1.

p =	*GUID prefixes and corresponding nodehandles n*															
0	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
	n	*n*	*n*	*n*	*n*	*n*		*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*
1	*60*	*61*	*62*	*63*	*64*	*65*	*66*	*67*	*68*	*69*	*6A*	*6B*	*6C*	*6F*	*6E*	*6F*
	n	*n*	*n*	*n*	*n*		*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*
2	*650*	*651*	*652*	*653*	*654*	*655*	*656*	*657*	*658*	*659*	*65A*	*65B*	*65C*	*65D*	*65E*	*65F*
	n	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*		*n*	*n*	*n*	*n*	*n*
3	*65A0*	*65A1*	*65A2*	*65A3*	*65A4*	*65A5*	*65A6*	*65A7*	*65A8*	*65A9*	*65AA*	*65AB*	*65AC*	*65AD*	*65AE*	*65AF*
	n		*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

- *n* represents [GUID, IP address] pairs that act as node handles specifying the next hop to be taken by messages addressed to GUIDs that match each given prefix.
- Grey-shaded entries in the table body indicate that the prefix matches the current GUID up to the given value of p

Pastry Routing Algorithm

- The routing process at any node *A* uses the information in its routing table *R* and leaf set *L*

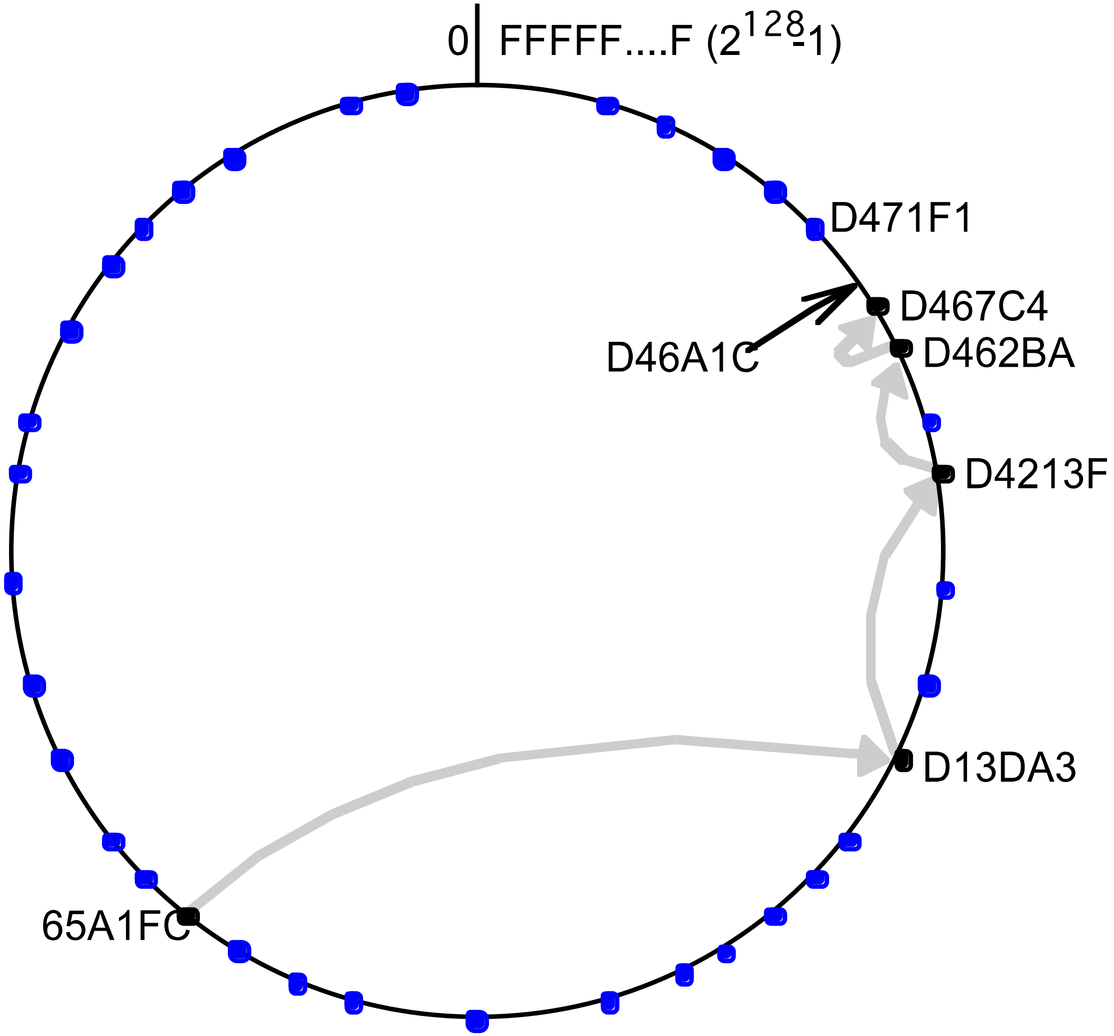
To handle a message *M* addressed to a node *D* (where $R[p,i]$ is the element at column *i*, row *p* of the routing table):

1. *If* ($L_{-l} < D < L_l$) { // the destination is within the leaf set or is the current node.
2.   Forward *M* to the element L_i of the leaf set with GUID closest to *D* or the current node *A*.
3. } *else* { // use the routing table to despatch *M* to a node with a closer GUID
4.   Find *p*, the length of the longest common prefix of *D* and *A*,. and *i*, the $(p+1)^{th}$ hexadecimal digit of *D*.
5.   *If* ($R[p,i] \neq null$) forward *M* to $R[p,i]$ // route *M* to a node with a longer common prefix.
6.   else { // there is no entry in the routing table.
7.   Forward *M* to any node in *L* or *R* with a common prefix of length *p* but a GUID that is numerically closer.

    }

}

Pastry routing example



Peer-to-peer systems
Summary

Comparison of discussed algorithms

PsP system	Model	Parameters	Hops to locate data	Routing state	Peers joins and leaves	Reliability
Napster	Centralized metadata index; Location inquiry from central server; Download directly from peer	None	Constant	Constant	Constant	Central server returns multiple download locations; client can retry
Gnutella	Broadcast request to as many peers as possible, download directly	None	no guarantee	Constant (approx 3-7)	Constant	Receive multiple replies from peers with available data; requester can retry
Pastry	Plaxton-style global mesh	N – number of peers in network b – base of the chosen identifier	$\log_b N$	$\log_b N$	logN	Replicate data across multiple peers; Keep track of multiple paths to each peer

What have we discussed today?

- We discussed different approaches to realize peer-to-peer systems. The earliest representative was pretty close to the c/s architecture.

- The approaches can be differentiated concerning the network structure and centrality. Starting here, we can explain the three/four examples and their general differences.

- We had a short introduction into overlay networks and how they are used.

- The concept of distributed hash tables in the context of structured peer-to-peer systems has been described and we are now able to explain it.

Questions

- What is a peer-to-peer system and what are its key characteristics? Discriminate the P2P paradigm from the client-server paradigm.
- Name typical application domains of P2P systems.
- What is an overlay network. (also see lecture about publish-subscribe systems).
- Name the main differences between 1st generation centralized P2P and pure P2P and 2nd generation hybrid P2P and DHT P2P systems.
- Briefly explain and compare the algorithms used for the placement of resources (e.g. data, files, ..) in each of this P2P system approaches.

References

George Coulouris, Jean Dollimore, Tim Kindberg: *Distributed Systems: Concepts and Design*. 5th edition, Addison Wesley, 2011.

Ripeanu, M., & Foster, I. (2002). Mapping the Gnutella Network: Macroscopic Properties of Large-Scale Peer-to-Peer Systems. PeerToPeer Systems First International Workshop IPTPS 2002 Cambridge MA USA March 78 2002 Revised Papers, 2429, 85-93. Springer. Retrieved from http://www.cs.rice.edu/Conferences/IPTPS02/128.pdf

J. Eberspächer, R. Schollmeier: First and Second Generation Peer-to-Peer Systems, In: R. Steinmetz, K. Wehrle (eds.): Peer-to-Peer Systems and Applications, Springer LNCS 3485, 2005

Stephanos Androutsellis-Theotokis and Diomidis Spinellis. 2004. A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. 36, 4 (December 2004), 335-371. DOI=10.1145/1041680.1041681

Karl Aberer, Luc Onana Alima, Ali Ghodsi, Sarunas Girdzijauskas, Seif Haridi, and Manfred Hauswirth. 2005. The Essence of P2P: A Reference Architecture for Overlay Networks. In Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P '05). IEEE Computer Society, Washington, DC, USA, 11-20. DOI=10.1109/P2P.2005.38

A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. IFIP/ACM Middleware 2001, Heidelberg, Germany, Nov. 2001