Institute of Computer Science
Department of Mathematics and Computer Science

Freie Universität Berlin

Algorithms and Programming IV

Peer-to-peer Computing and Networking

Summer Term 2023 | 28.06.2023

Claudia Müller-Birn, Barry Linnert

Recap

- Differences RMI and RPC
- Main Components of RMI
- Simple example Java RMI

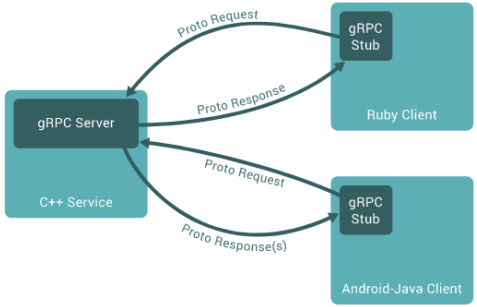
An Addition to RMI

Is it possible to realize RMI in Python?

- Yes. Check out PYthon Remote Objects (Pyro)
- Further information: https://pyro4.readthedocs.io/en/stable/

Are there other alternatives?

- Yes. Check out gRPC, which is an open source RPC framework
- Further information: https://grpc.io/



Our Topics Today

- Motivation Peer-to-Peer (P2P) Systems
  - Overlay Networks
- Unstructured Overlay Networks
  - Centralized P2P Systems
  - Pure P2P Systems
  - Mixed P2P Systems
- Structured Overlay
  - DHT-based P2P Systems

Alternative organizations

Vertical distribution

- It comes from dividing distributed applications into three logical layers and running the components from each layer on a different server (machine).

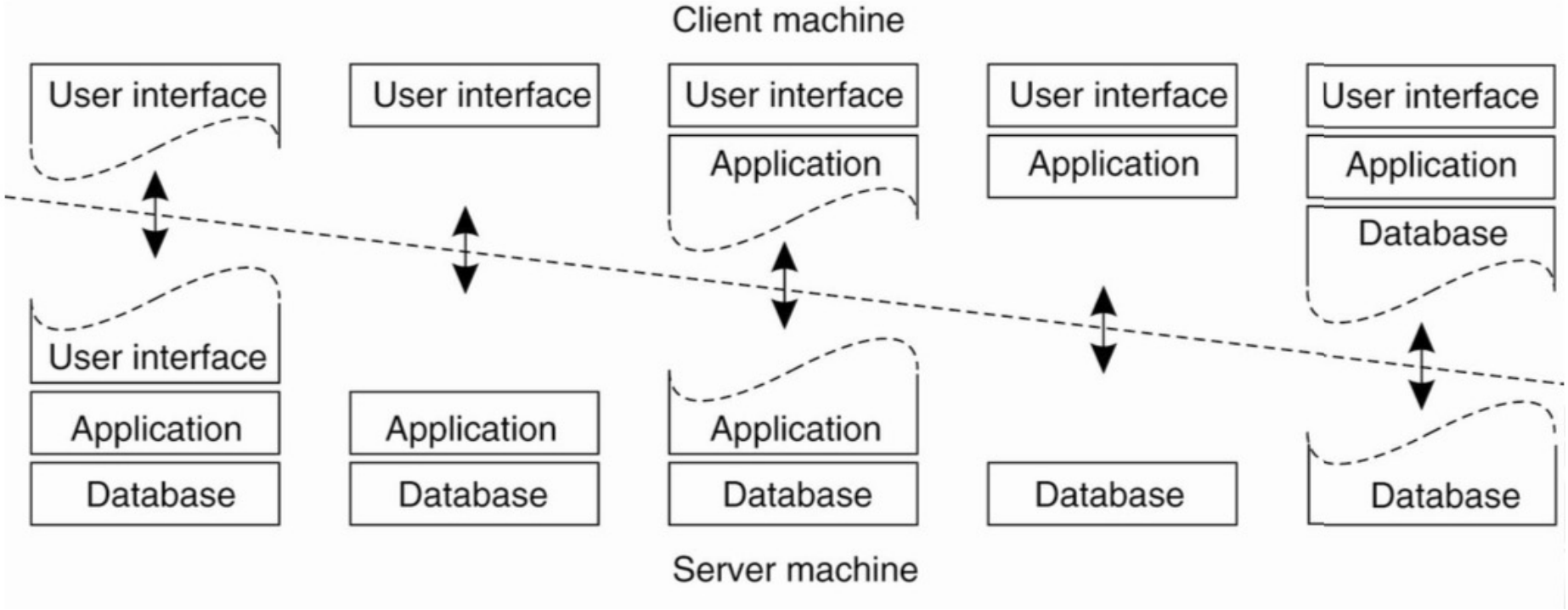
Horizontal distribution

- A client or server may be physically split up into logically equivalent parts, but each part operates on ist own share of the complete data set.

Peer-to-peer architectures

- Processes are all equal: the functions that need to be carried out are represented by every process ⇒ each process will act as a client and a server simultaneously (i.e., acting as a servant).

Possible vertical distribution



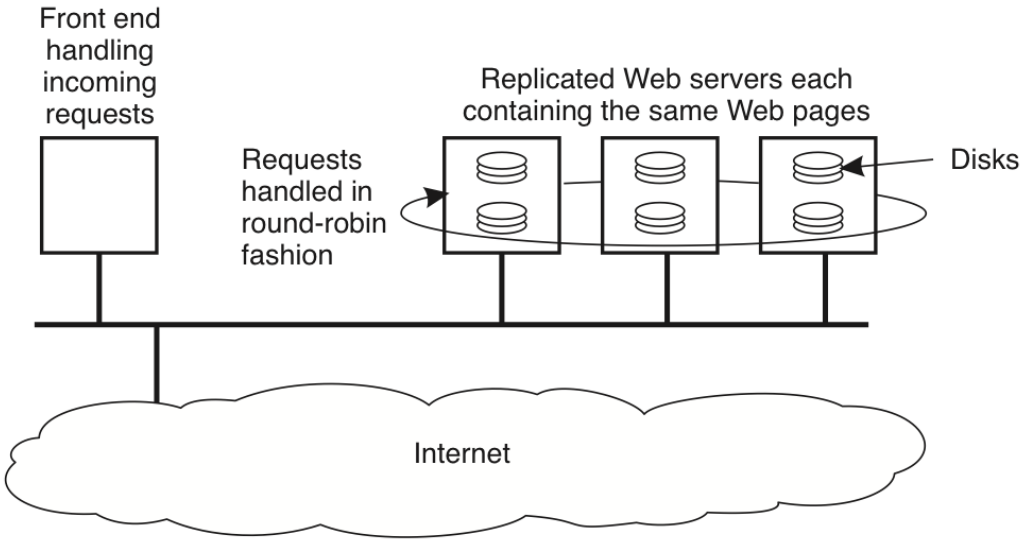
Tanenbaum & van Steen. Distributed Systems. Principles and Paradigms. 2007.

Possible horizontal distribution

Involves replicating a server's functionality over multiple computers to improve scalability (by reducing the load on individual servers) and reliability (by providing redundancy).

Replicated Web server

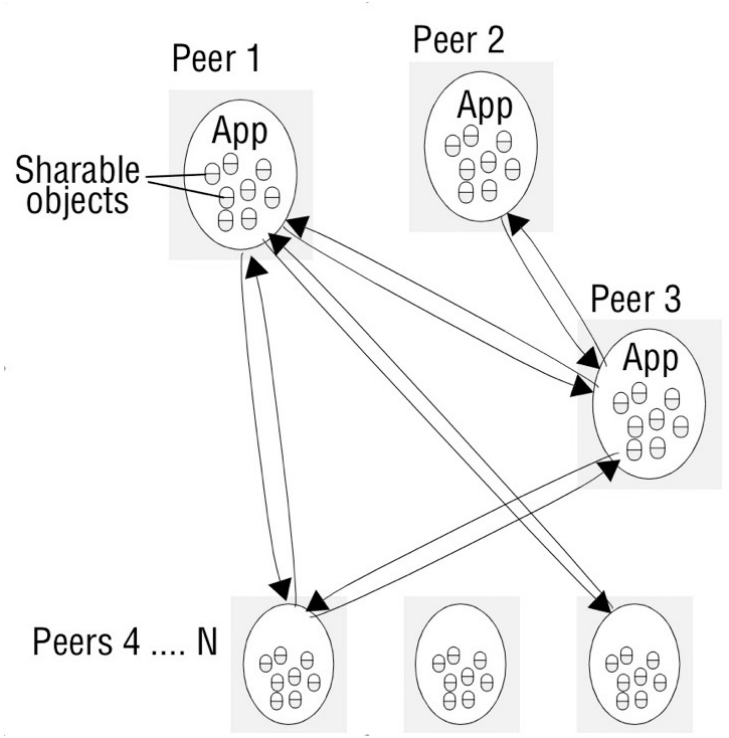
- Each server machine contains a complete copy of all hosted Web pages.
- Client requests are passed on to the servers in a round robin fashion.



Coulouris, Dollimore, Kindberg: *Distributed Systems: Concepts and Design*. 2011.

Peer-to-Peer Architecture

- Is composed of a large number of peer processes running on separate computers
- All processes have client and server roles: *servent*
- Patterns of communication between them depends entirely on application requirements
- Need to place and retrieve individual computers is more complex then in client-server architecture



Motivation Peer-to-Peer (P2P) Architectures

- Represent a paradigm of distributed systems and applications in which data and computational resources are contributed by many hosts on the Internet.
- Enable the sharing of data and resources on a large scale by eliminating any requirement for separately managed servers and their associated infrastructure.
- Used to provide file sharing, web caching, information distribution, and other services.
- Peer-to-peer architectures revolve around how to organize the processes in an overlay network.

Overlay Networks

An *overlay network* consists of nodes and virtual links, which sit on top of an underlying network (such as an IP network).

An overlay network offers something that is not otherwise provided:

- a service that is tailored towards the needs of a class of application or a particular higher-level service (e.g., multimedia content distribution);
- more efficient operation in a given networked environment (e.g. routing in an ad hoc network);
- an additional feature (e.g. secure communication via VPN).

Advantages and Disadvantages of Overlay Networks

(+)

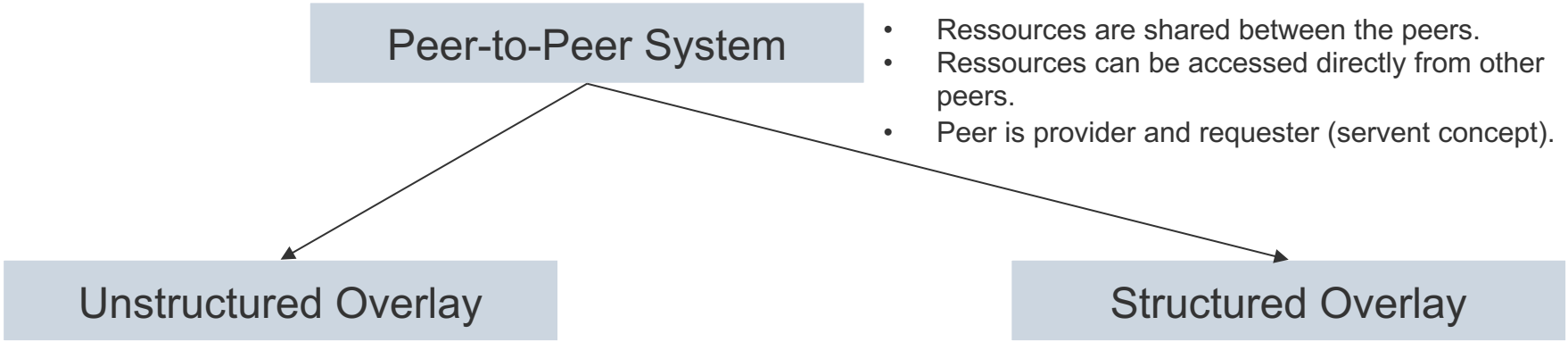
- They enable new network services to be defined without requiring changes to the underlying network.
- They encourage experimentation with network services and the customization of services.
- Multiple overlays can be defined and can coexist.

(-)

- Introduce an extra level of indirection
- Add to the complexity of network services

Application	Service Description	Examples
Content Distribution	Scalable approaches to sharing data with users across the Internet	• File storage and sharing: Gnutella, Bittorrent, Seti@home • Content delivery networks (CDNs): Akamai, Limelight • Streaming media: Spotify, Sonos • Software update distribution: Linux, World of Warcraft
Distributed Computing	Delegating the work for an application across many computers	• Privacy and censorship resistance: Tor, Freenet • Cryptocurrency: Bitcoin • Botnets and malware: Storm
Collaboration	Providing real-time human communication	• Voice Over IP (VOIP): Skype (originally) • Instant Messaging: Tox
Platforms	Building applications	• Java: JXTA

https://w3.cs.jmu.edu/kirkpams/OpenCSF/Books/csf/html/AppLayer.html



- Ressources are shared between the peers.
- Ressources can be accessed directly from other peers.
- Peer is provider and requester (servent concept).

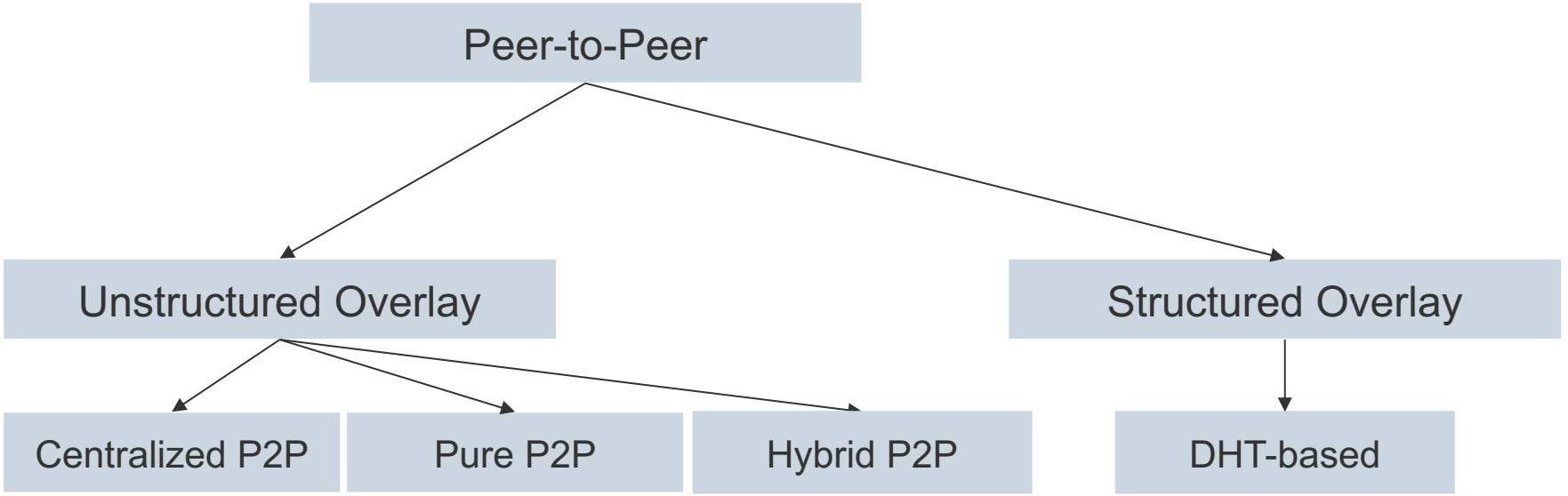
Characteristics of Unstructured Overlay Network

The **placement** of content (files) is completely unrelated to the overlay topology and the content typically needs to be located.

- Peers & resources have no special identifier.
- Each peer is responsible only for the resources it submitted.
- The location of the resources is only known to submitter.
- New resources can be introduced at any location.

The main **task** in such networks is to search

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


Peer-to-Peer
Unstructured Overlay
Structured Overlay
Centralized P2P
Pure P2P
Hybrid P2P
DHT-based

Peer-to-Peer

- Ressources are shared between the peers.
- Resoources can be accessed directely from other peers.
- Peer is provider and requester (servent concept).

Unstructured Overlay

Structured Overlay

Centralized P2P

1st Generation

- All features pf P2P included
- Central entity is necessary to provide the service
- Central entity is some kind of index/group database
- Example: Napster

Peer-to-peer systems

NAPSTER

Introducing Napster

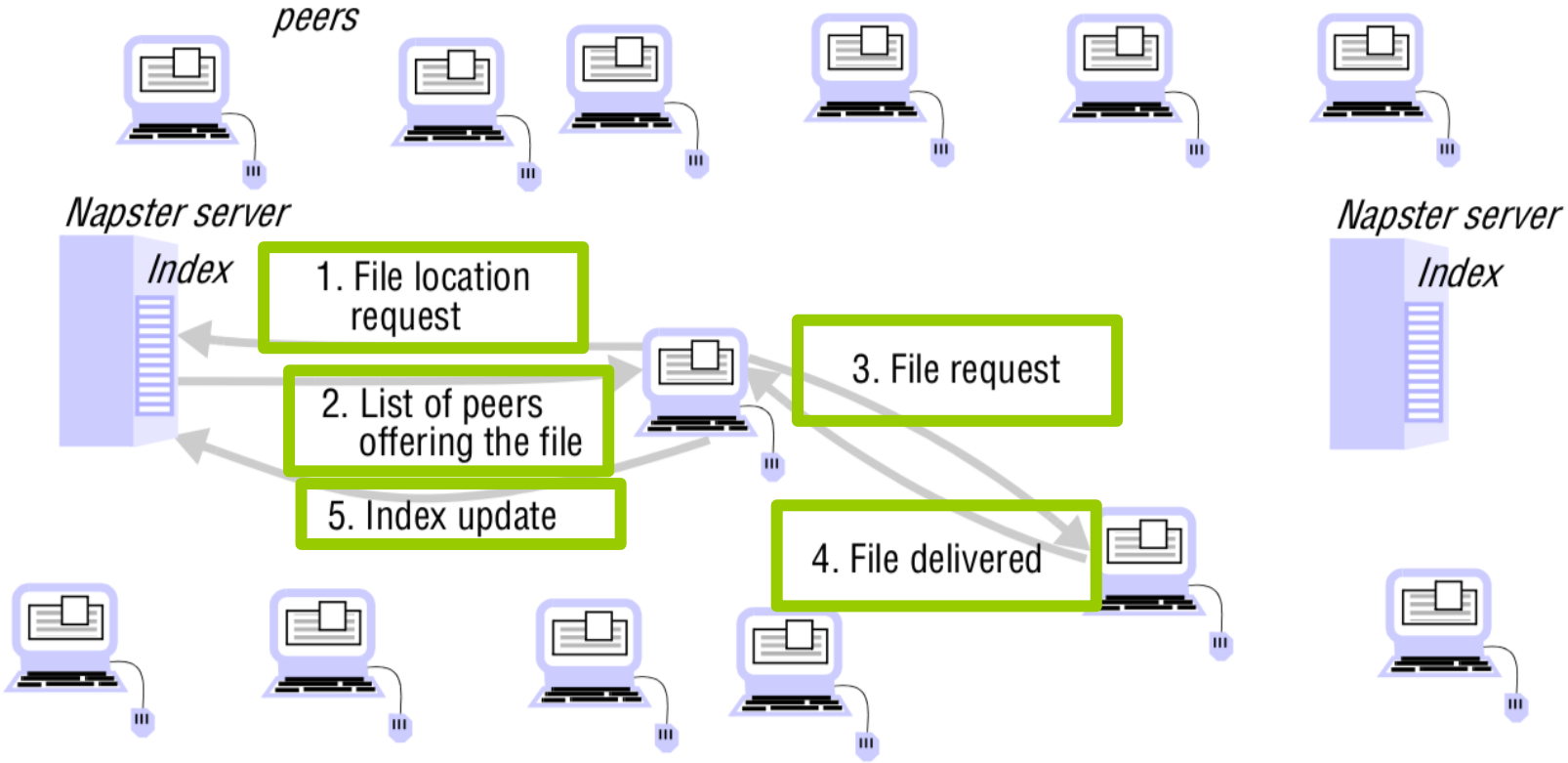
In June 1999, the first peer-to-peer file sharing system, Napster was released.

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

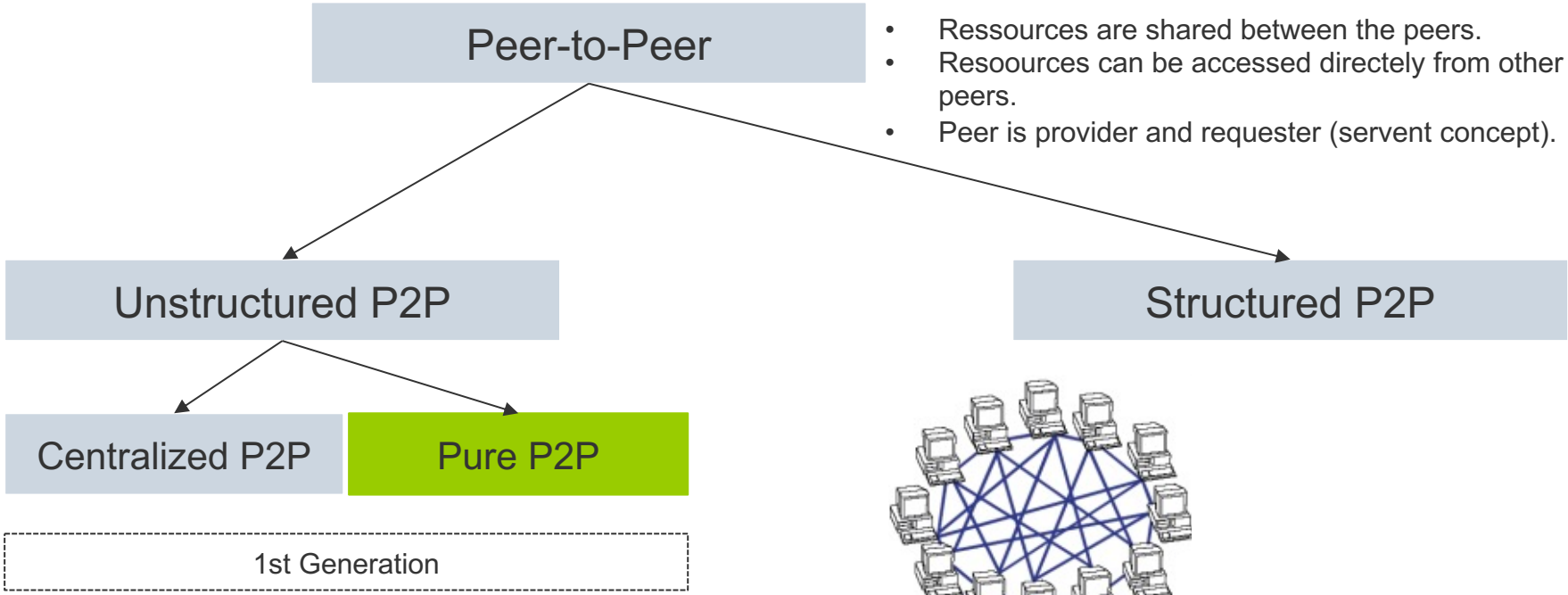
Napster took advantage of special characteristics of the application: music files are never updated, no guarantees are required concerning the availability of individual files.

Advantages

- Simple to implement and files are located quickly and efficiently.

Disadvantages

- Vulnerable to censorship, legal action, surveillance, malicious attack, and technical failure, since it is controlled by a central server.
- 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.



- Ressources are shared between the peers.
- Resoources can be accessed directely from other peers.
- Peer is provider and requester (servent concept).

- All features pf P2P included
- Any terminal entity can be removed without loss of functionality
- No central entity
- Example: Gnutella 0.4, Freenet

Peer-to-peer systems

GNUTELLA 0.4

Introducing Gnutella

There is no overall control over the topology or the placement of nodes in the network. Nodes connect to each other directly in a ad-hoc fashion.

Similarities between Gnutella and Napster

- Nodes 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.
- Nodes 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 Incredients

Gnutella is a protocol for peer-to-peer search, consisting of:

- A set of **message** formats
  - 5 basic message types
- A set of **rules** governing the exchange of messages
  - Broadcast
  - Back-propagate
  - Handshaking
- An hostcache for node bootstrap.

Gnutella Messages

Each message is composed of:

- A 16-byte ID field uniquely identifying the message randomly generated not related to the address of the requester (anonymity) used to detect duplicates and route back-propagate messages.
- A message type field
  - `ping`, `pong`,
  - `query`, `queryhit`
  - `push` (for firewalls)
- A Time-To-Live (TTL) Field
- Payload length

Gnutella Messages (*cont.*)

`Ping` (broadcast)

- Used to maintain information about the nodes currently in the network
- Originally, a “who’s there” flooding message
- A peer receiving a `ping` is expected to respond with a `pong` message

`Pong` (back-propagate)

- A `pong` message has the same ID of the corresponding `ping` message
- Contains:
  - address of connected Gnutella peer
  - total size and total number of files shared by this peer

Gnutella Messages (*cont.*)

`Query` (broadcast)

- The primary mechanism for searching the distributed network
- Contains the query string
  - A servent is expected to respond with a `queryhit` message if a match is found against its local data set

`Queryhit` (back-propagate)

- The response to a query
- Has the same ID of the corresponding `query` message
- Contains enough information to acquire the data matching the corresponding query (IP Address + port number, List of file names)

Phases of the Protocol 0.4

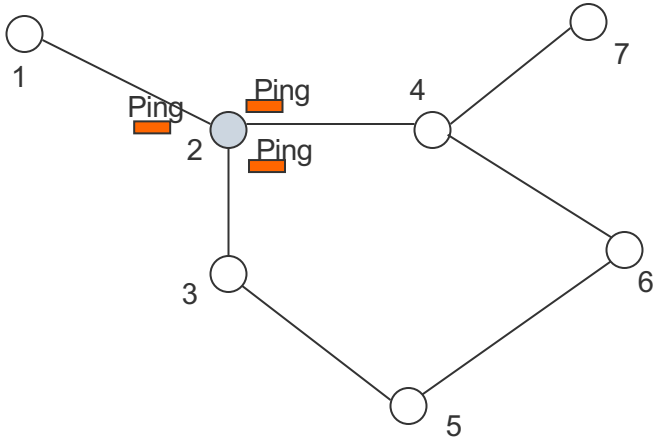
1. Membership / Connect
   - **`Ping`**: initiating message (“I’m here”)
   - **`Pong`**: reply to a ping, contains information about the peer

2. Search
   - **`Query`**: search pattern and TTL (time-to-live)
   - **`Query` response**: contains information about the computer that has the needed file

3. Data Transfer (HTTP is used to transfer files)
   - **`GET`**: return the requested file
   - **`PUSH`**: push the file to me (to circumvent firewalls)

(Horowitz 2002)

Gnutella Connect Mechanism

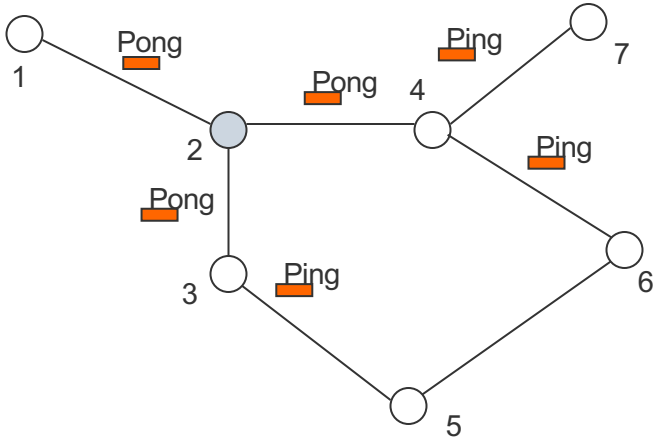


Steps:

1. Node 2 connect to network

(Horowitz 2002)

Gnutella Connect Mechanism

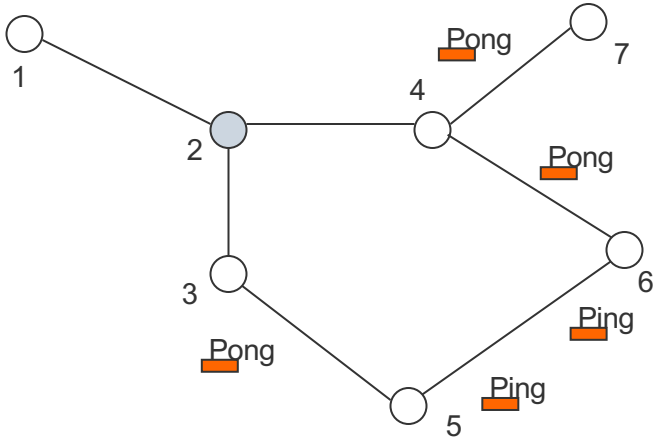


Steps:

1. Node 2 connect to network via `ping`
2. Node 1, 3 and 4 reply via `pong`

(Horowitz 2002)

Gnutella Connect Mechanism

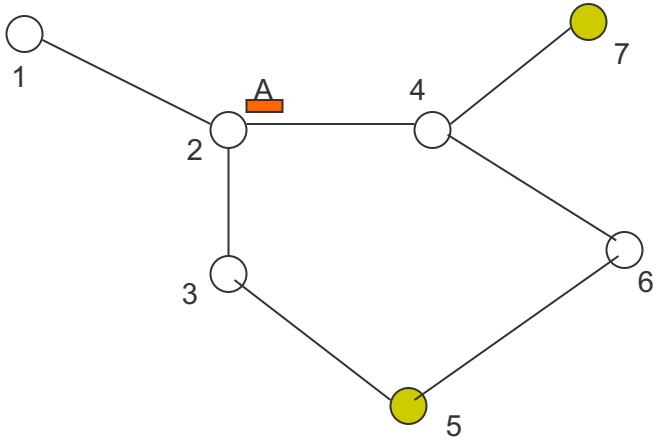


Steps:

1. Node 2 connect to network via `ping`
2. Nodes 1, 3 and 4 reply via `pong`
3. Nodes 5, 6 and 7 reply via `pong`

(Horowitz 2002)

Gnutella Search Mechanism



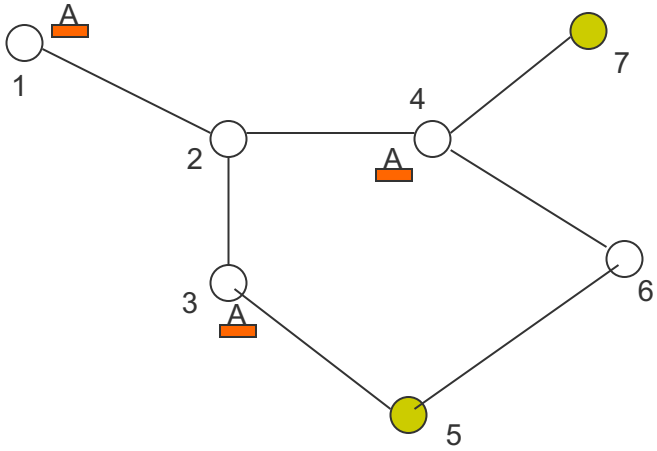
Steps:

1. Node 2 initiates search for file A

A node that receives a QUERY message increases the HOP count field of the message
IF HOP <= TTL (Time To Live) and a QUERY message with this GUID was not received before
THEN forwards it to all nodes except the one he received it from

(Horowitz 2002)

Gnutella Search Mechanism

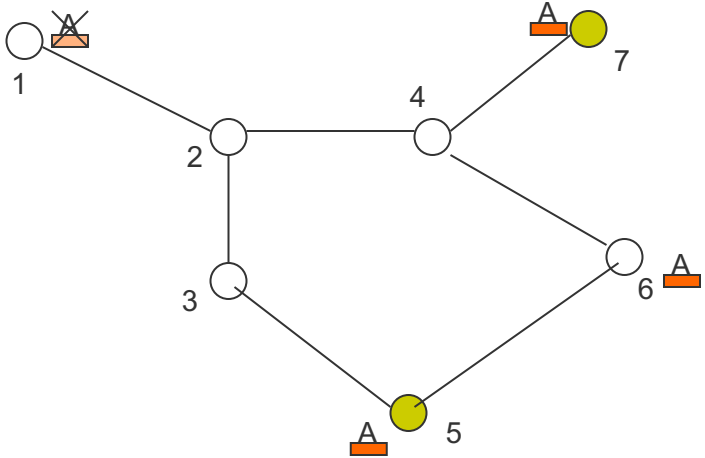


Steps:

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

(Horowitz 2002)

Gnutella Search Mechanism

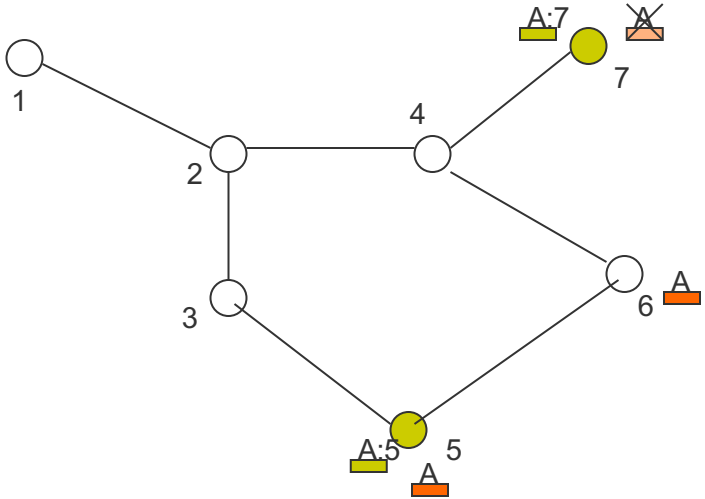


Steps:

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

(Horowitz 2002)

Gnutella Search Mechanism

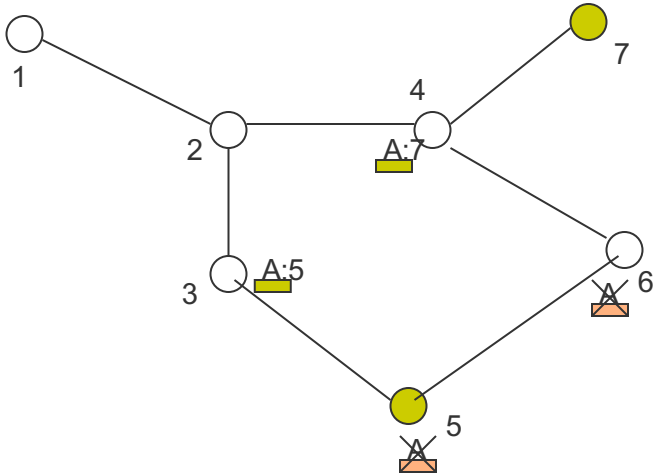


Steps:

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:

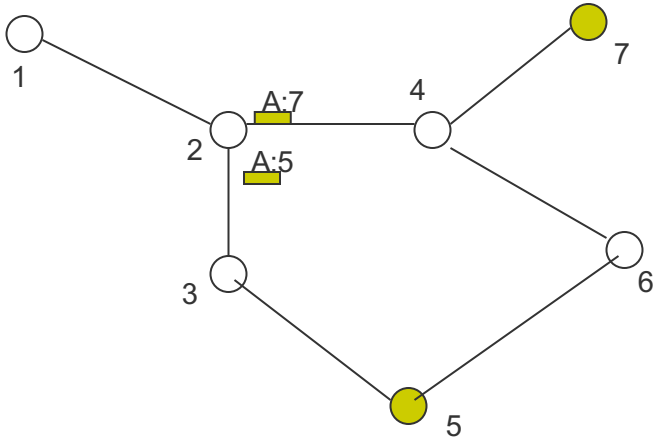
1. Node 2 initiates search for file A
2. Sends message to all neighbors
3. Neighbors forward message
4. Nodes that have file A initiate a reply message
5. Query reply message is back-propagated

The QUERYHIT message contains
- The IP address of the sender
- Information about one or more files that match search criteria

Information passed back the same way the QUERY took
- No flooding

(Horowitz 2002)

Gnutella Search Mechanism

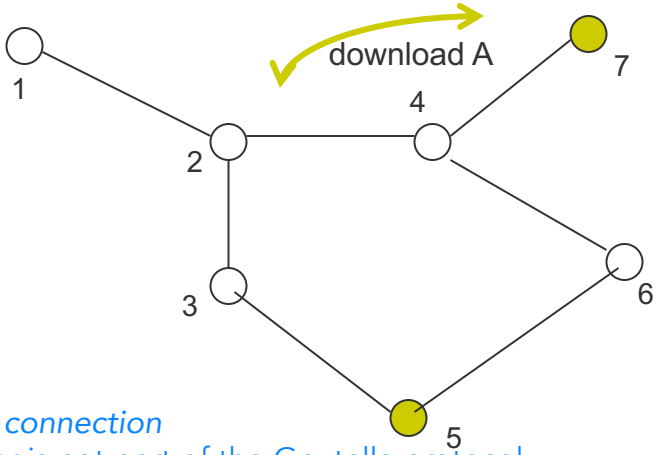


Steps:

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

Peer sets up a HTTP connection

- Actual data transfer is not part of the Gnutella protocol
- HTTP GET is used

Special case:

- Peer with the file located behind a firewall/NAT gateway
- Downloading peer
  - Cannot initiate a TCP/HTTP connection
  - Can instead send the PUSH message
    - Asking the other peer to initiate a TCP/HTTP connection to it and
    - Then transfer (push) the file via it
  - Does not work if both peers are behind firewalls

(Horowitz 2002)

Gnutella Search Strategy: Flooding

Breadth-first search (BFS) -> ALP3

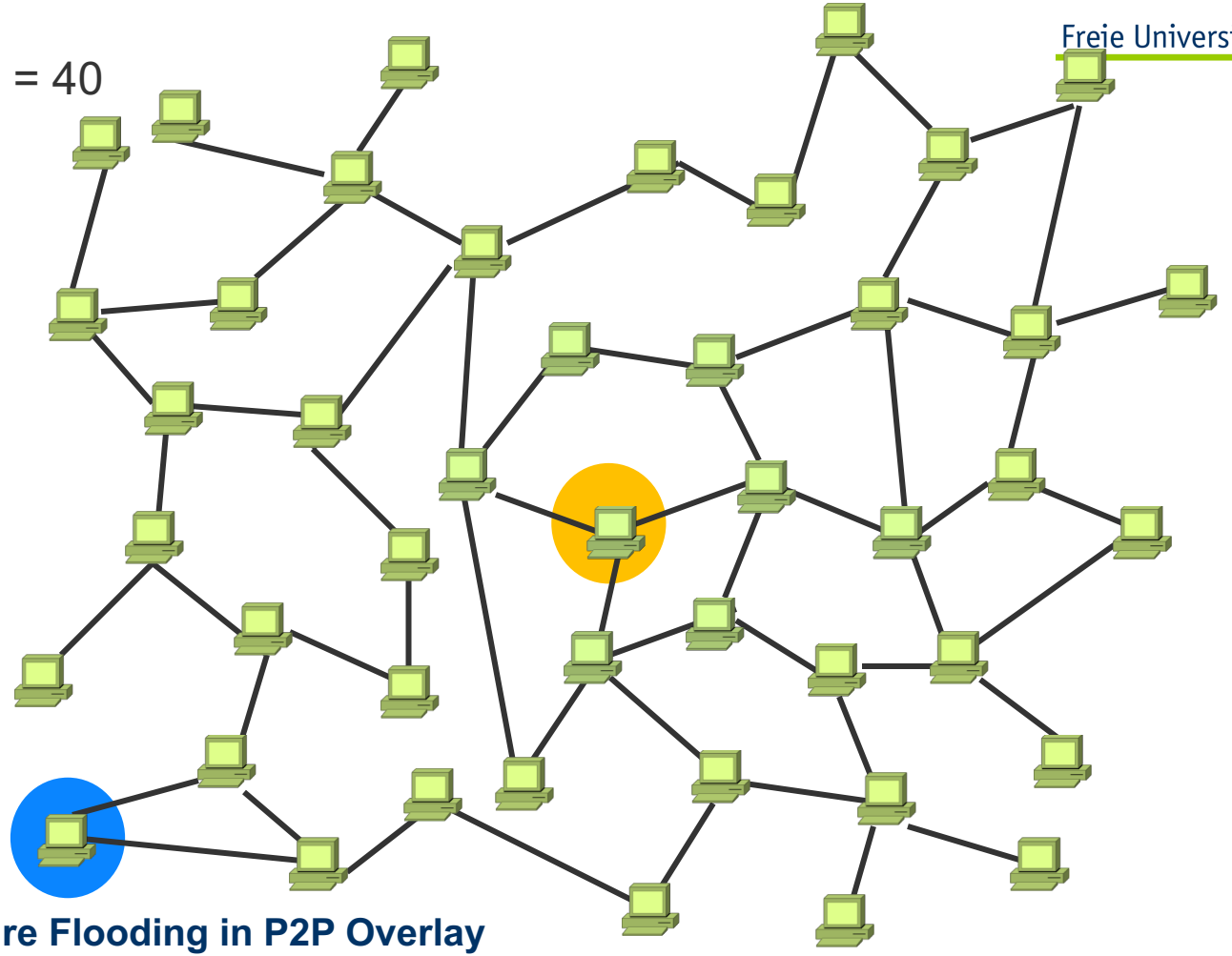
- Send a message to all neighbors, except the one who delivered the incoming message
- Use system-wide maximum Time-to-Live (TTL)
  - Decrement with every hop
  - Discard message if TTL negative
- Store message identifiers of routed messages to avoid retransmission cycles
- Highest effort for searching

Advantages

- Simple and robust and no state maintenance needed

HOPS = 4

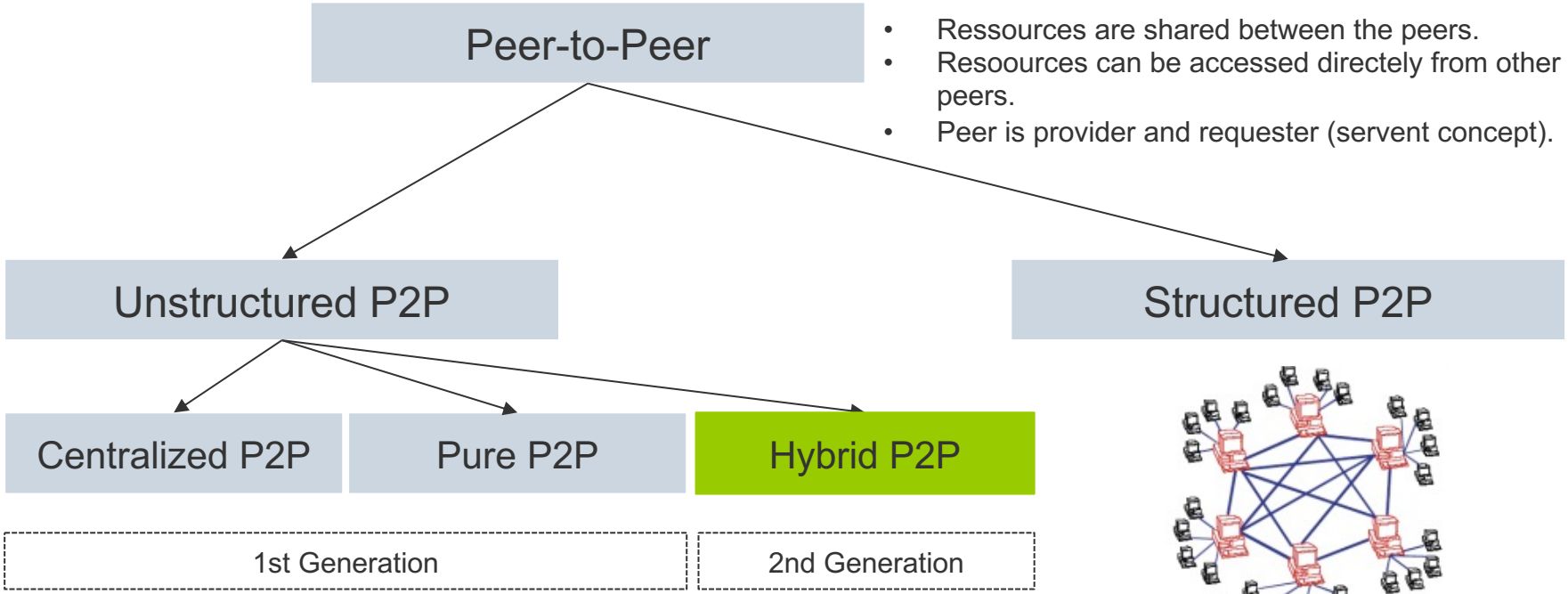
Messages = 40



Pure Flooding in P2P Overlay

Problems of Gnutella Search Strategy: Flooding

- Loops in Gnutella networks which are caused by redundant links and result in endless message routing. Solutions by Gnutella:
  - Detect and discard redundant messages (by their GUID)
  - Limit TTL (time-to-live) of messages (originally TTL = 7)
- Gnutella 0.4 suffers from a range of scalability issues, due to
  - Fully decentralized approach
  - Flooding of messages
- Low bandwidth peers easily use up all their bandwidth
  - For forwarding PING and QUERY
  - No bandwidth for up- and downloads
- Breakdown of Gnutella network in August 2000



- Ressources are shared between the peers.
- Resoources can be accessed directely from other peers.
- Peer is provider and requester (servent concept).

- All features pf P2P included
- Any terminal entity can be removed without loss of functionalit
- Dynamic central entities
- Example: Gnutella 0.6

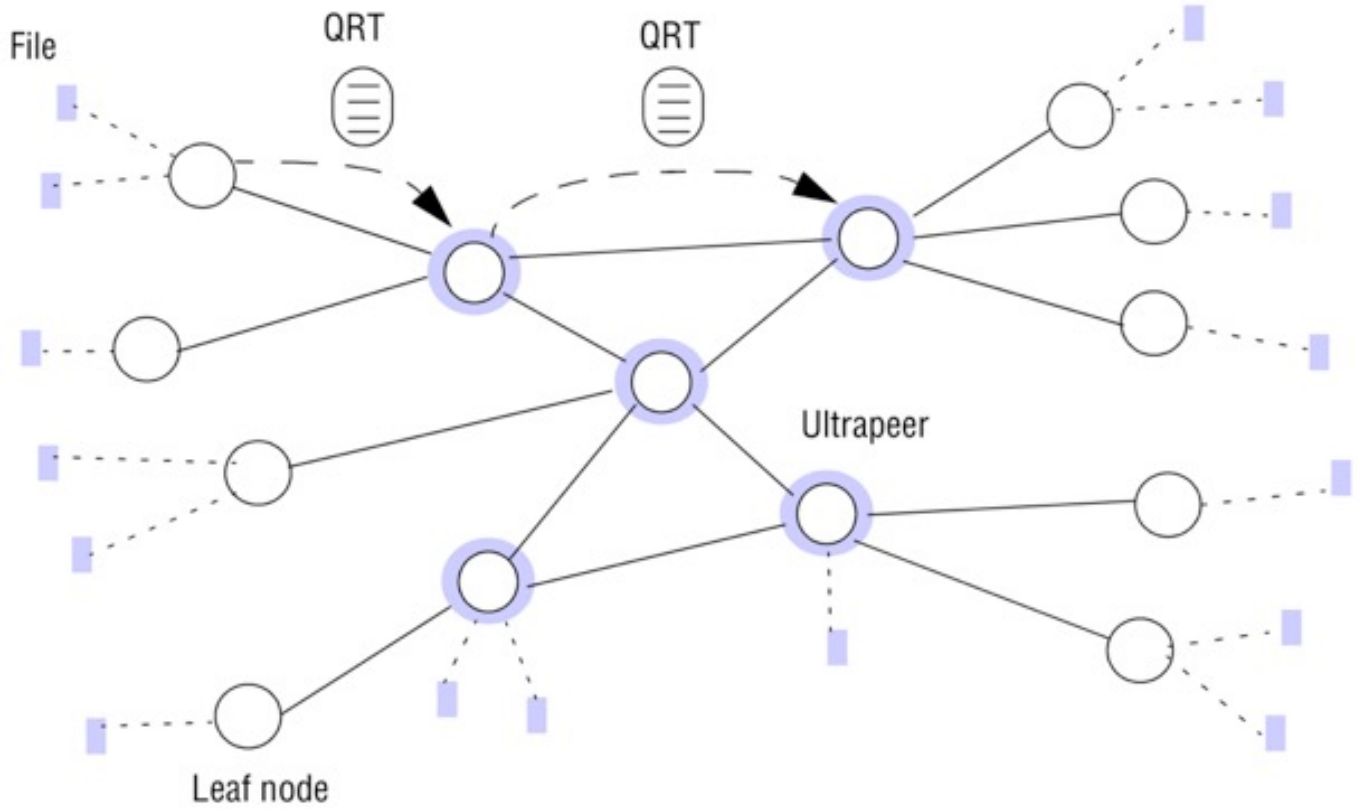
Peer-to-peer systems

GNUTELLA 0.6

Improvements of the New Protocol

- 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 files available on that node.

Key Elements in the Gnutella 2 Protocol



Peer-to-Peer

- Ressources are shared between the peers.
- Resoources can be accessed directely from other peers.
- Peer is provider and requester (servent concept).

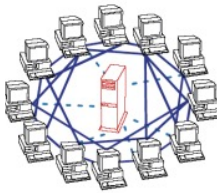
Unstructured Overlay

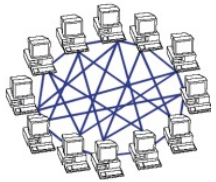
Structured Overlay

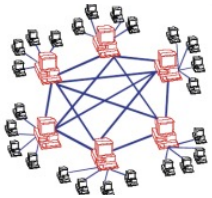
Centralized P2P

Pure P2P

Hybrid P2P







Characteristics of Structured Overlay Network

The overlay **topology** is tightly controlled and files (or pointers to them) are placed at precisely specified locations.

- 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
- 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.

Peer-to-Peer

- Ressources are shared between the peers.
- Resoources can be accessed directely from other peers.
- Peer is provider and requester (servent concept).

Unstructured Overlay

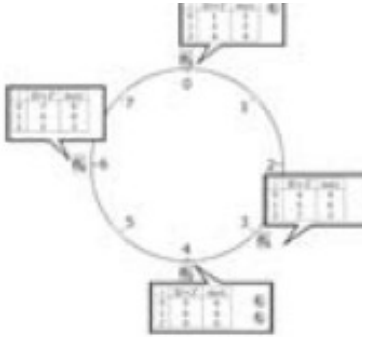
Centralized P2P

Pure P2P

1st Generation

Structured Overlay

DHT-based



- All features pf P2P included
- Any terminal entity can be removed without loss of functionality
- No central entities
- Connections in the overlay are fixed
- Example: Paestry, Chord, CAN

Peer-to-peer systems

PASTRY

Introducing Pastry

- P2P overlay that is using Dynamic 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 after each hop.

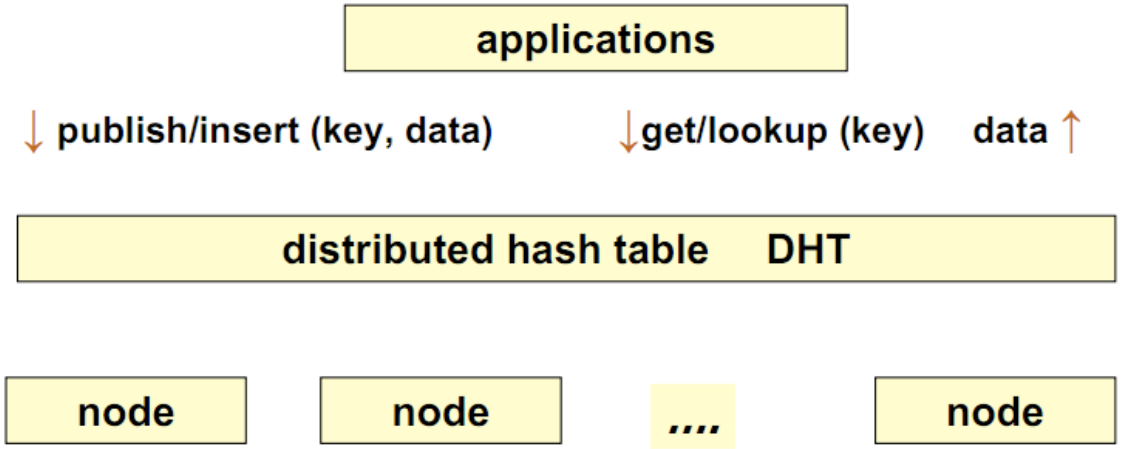
- It is originally developed Microsoft and Rice Uni but a free version (FreePastry) exists that is a prototypical implementation of Pastry. The latter is mostly used by scientific community.

- Similar algorithms are Chord and CAN.

Introducing Pastry (*cont.*)

- Any computer connected to the Internet and running PASTRY node software can be a PASTRY node.
- Application specific security polices may be applied.
- Each node is identified by a unique 128 bit node identifier (NodeId).
  - The node identifier is assumed to be generated randomly
  - Each NodeId in is assumed to have the same probability of being chosen
  - Node with similar NodeId may be geographically far

Mode of Operation of a Distributed Hash Table



lookup(key) -> node or data directly

Lookup where it is stored

every node stores and maintains part of hash table

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

Distributed Hash Table: Steps of Operation

1. Mapping of nodes and data same address space
   - Peers and content are addressed using flat identifiers (IDs)
   - Common address space for data and nodes
   - Nodes are responsible for data in certain parts of the address space
   - 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
   - Responsible node not necessarily known in advance
   - Deterministic statement about availability of data

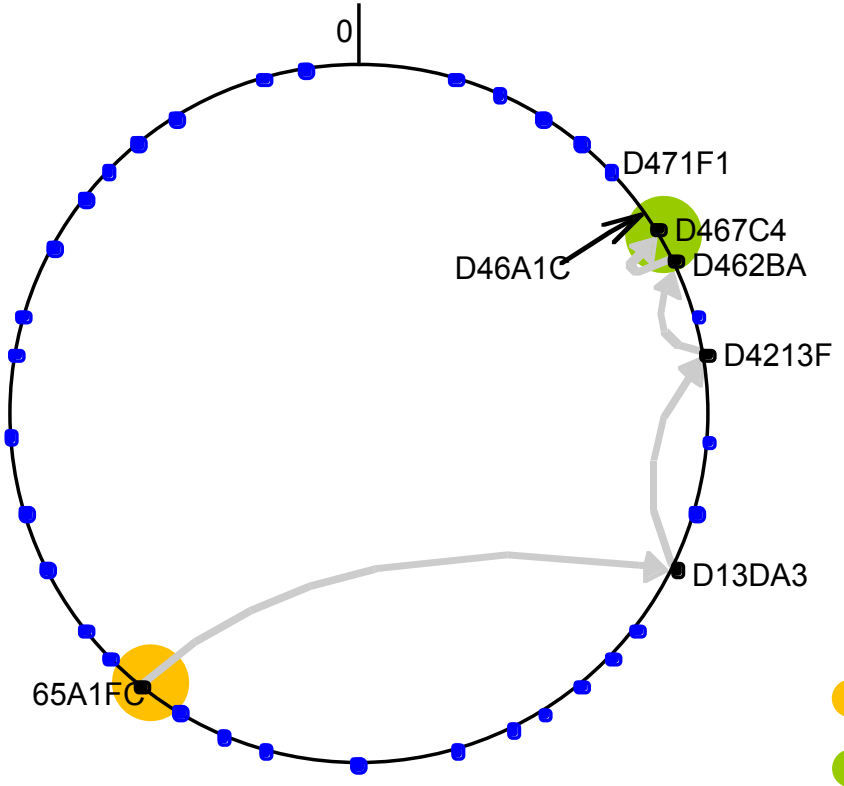
Sketch of the Routing Algorithm

- Assume we want to find the node in the PASTRY network with the NodeId closest to a given key.

Routing Idea

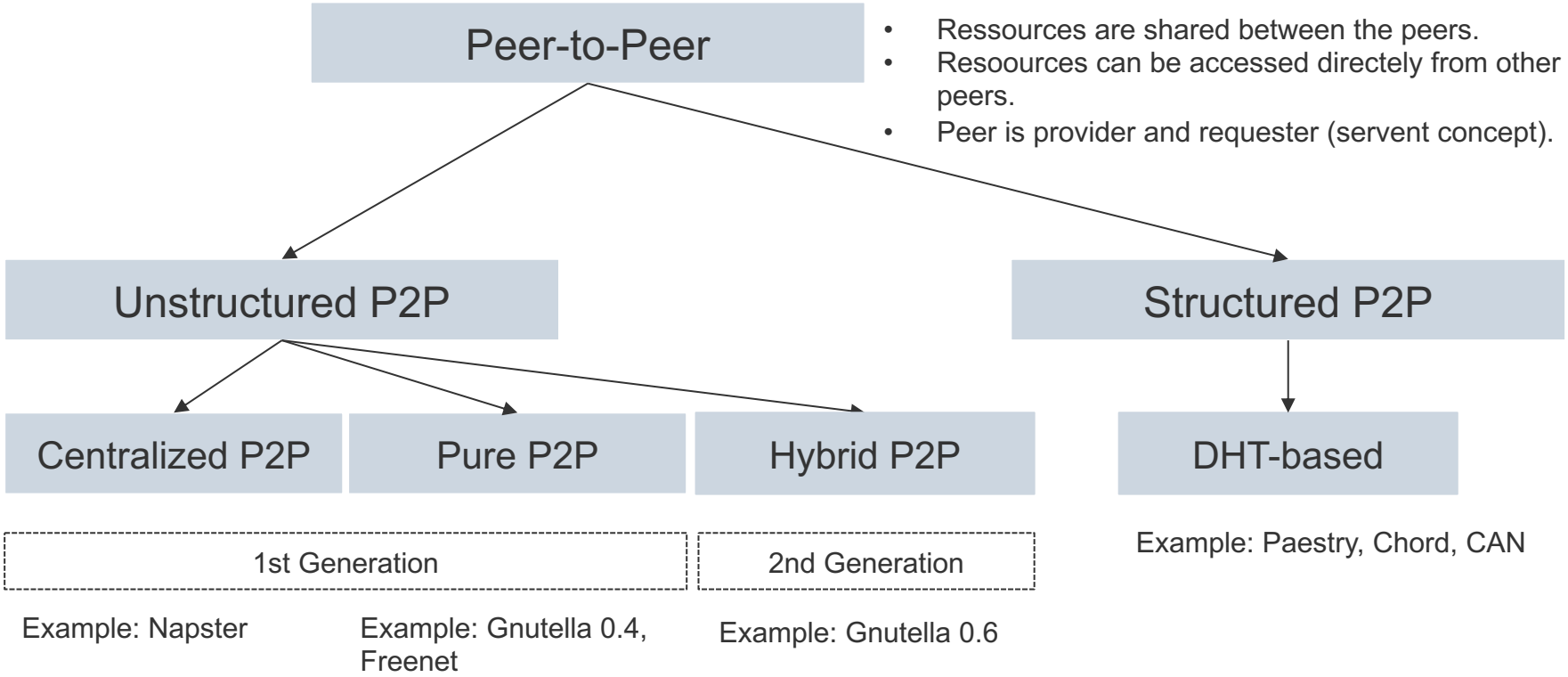
In each routing step, a node normally forwards the message to a node whose NodeId shares with the key a prefix that is at least one digit longer than than the key shares with the present node. If such a node is not known, the message is forwarded to a node that shares the same prefix of the actual node but its NodeId is numerically closer to the key.

Pastry Routing Example



Source NodeID **65A1FC**

Destination NodeID **D46A1C**



- Ressources are shared between the peers.
- Resoources can be accessed directely from other peers.
- Peer is provider and requester (servent concept).

Structured vs. Unstructured P2P Systems

	Unstructured peer-to-peer	Structured peer-to-peer
Advantages	Self-organizing and naturally resilient to node failure.	Guaranteed to locate objects (assuming they exist) and can offer time and complexity bounds on this operation; relatively low message overhead.
Disadvantages	Probabilistic and hence can not offer absolute guarantees on locating objects; prone to excessive messaging overhead which can affect scalability.	Need to maintain often complex overlay structures, which can be difficult and costly to achieve, especially in highly dynamic environments.

Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design

Summary

- In peer-to-peer, clients and servers are not differentiated. In client-server, a centralized server is used to store the data, while in peer-to-peer systems, each peer has its own data.
- We discussed different applications for peer-to-peer systems.
- We discussed different approaches to realizing peer-to-peer systems based on their overlay topology.
- In peer-to-peer systems, every node can do both request and respond to the services.
- Peer-to-peer systems are less stable if the number of peers is increasing.

Addition: Bitcoin

An introduction on a very general level, by Neha Narula (TED Talk):

- https://www.youtube.com/watch?v=pPgd7Hj3ABQ

A technical discussion of Bitcoin by 3Blue1Brown:

- https://www.youtube.com/watch?v=bBC-nXj3Ng4

A discussion of the technology of blockchain itself provided by Ittai Abraham:

- https://www.youtube.com/watch?v=iPRrGqDsMxg

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

Thanks to Prof. Florian Tschorsch for all these recommendations!

Institute of Computer Science
Department of Mathematics and Computer Science

Freie Universität Berlin

Algorithms and Programming IV

Cloud Computing

Summer Term 2023 | 03.07.2023

Claudia Müller-Birn, Barry Linnert

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
- Smruti R. Sarangi: Distributed Hash Tables Pastry (http://www.cse.iitd.ernet.in/~srsarangi/csl860/docs/pastry-lec.pdf)
- Marty Stepp: CSE 373: Data Structures and Algorithms, Winter 2013. University of Washington Computer Science & Engineering (https://courses.cs.washington.edu/courses/cse373/13wi/)