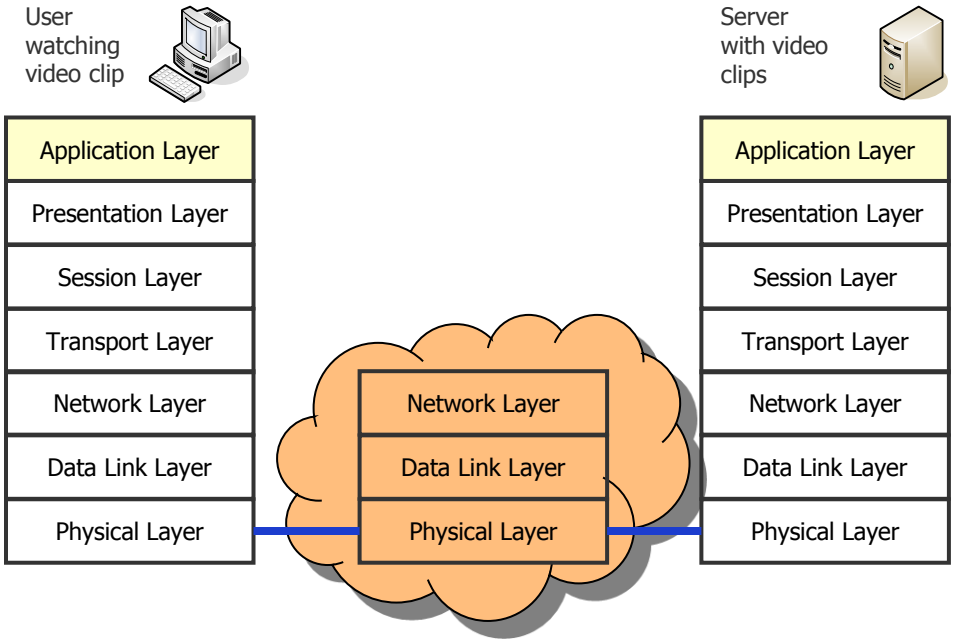
Telematics
Chapter 10: Peer-to-Peer Networks



Univ.-Prof. Dr.-Ing. Jochen H. Schiller

Computer Systems and Telematics (CST)

Institute of Computer Science

Freie Universität Berlin

http://cst.mi.fu-berlin.de

Contents

- Design Issues
- Basics of Peer-to-Peer systems
- First Generation P2P-Systems
  - Napster
  - Gnutella
  - Super-Peer-Networks
- Distributed Hash Tables
  - Concept
  - Case Study I: Chord
  - Case Study II: Pastry
- Overlay vs. physical topology

Design Issues

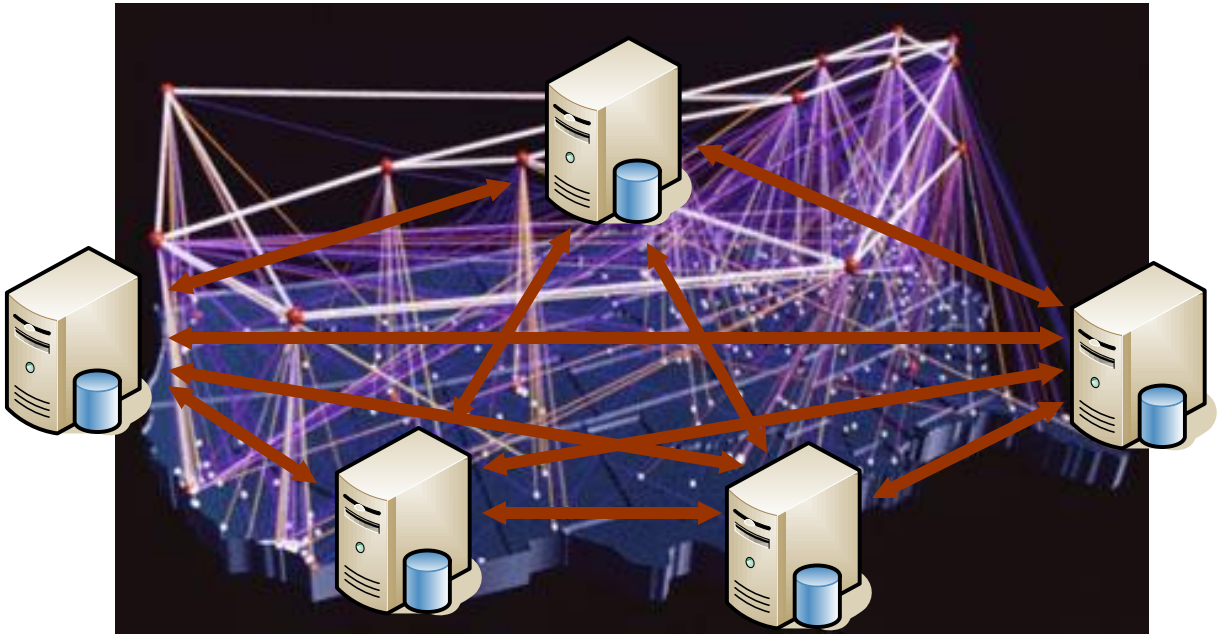
- Two communication principles
  - Client/Server (C/S)
  - Peer-to-Peer (P2P)
- Originally the Internet has a P2P design
  - All hosts provide services like
    - FTP, Telnet, Email
- With a growing network a more hierarchical network structure was established
  - Client/Server applications
- Since 2000 a new wave of P2P applications emerged

OSI Reference Model

Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

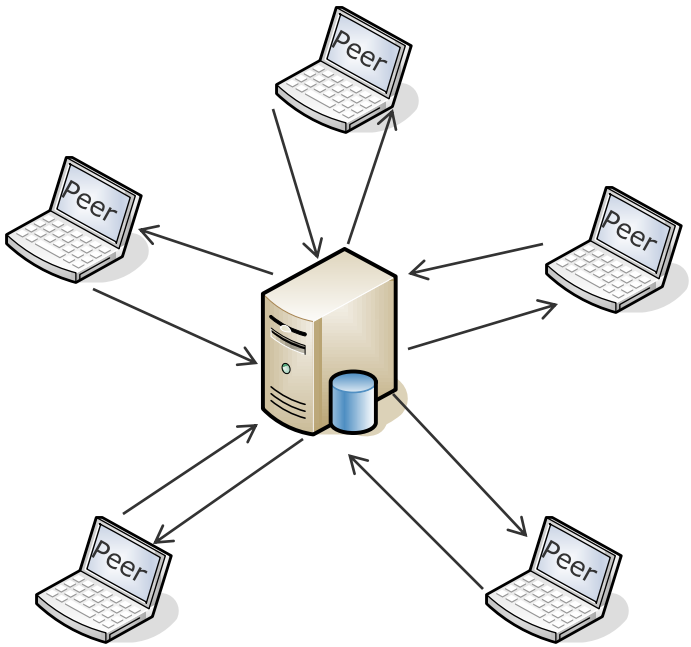
Peer-to-Peer: Introduction

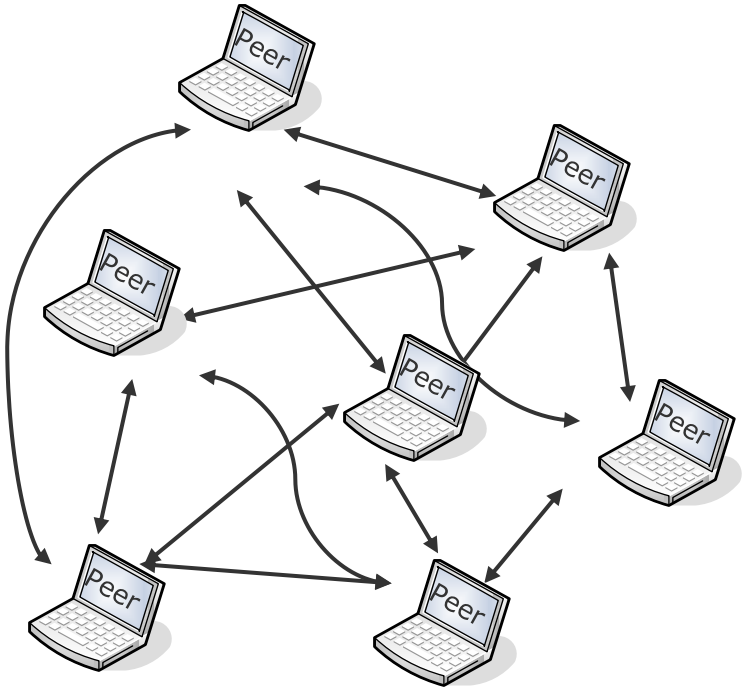
- Originally the Arpanet/Internet has a P2P design
  - All hosts provide services like (FTP, Telnet, …)
  - All hosts were equal
  - With increasing number of nodes the structure changed to a Client-Server architecture



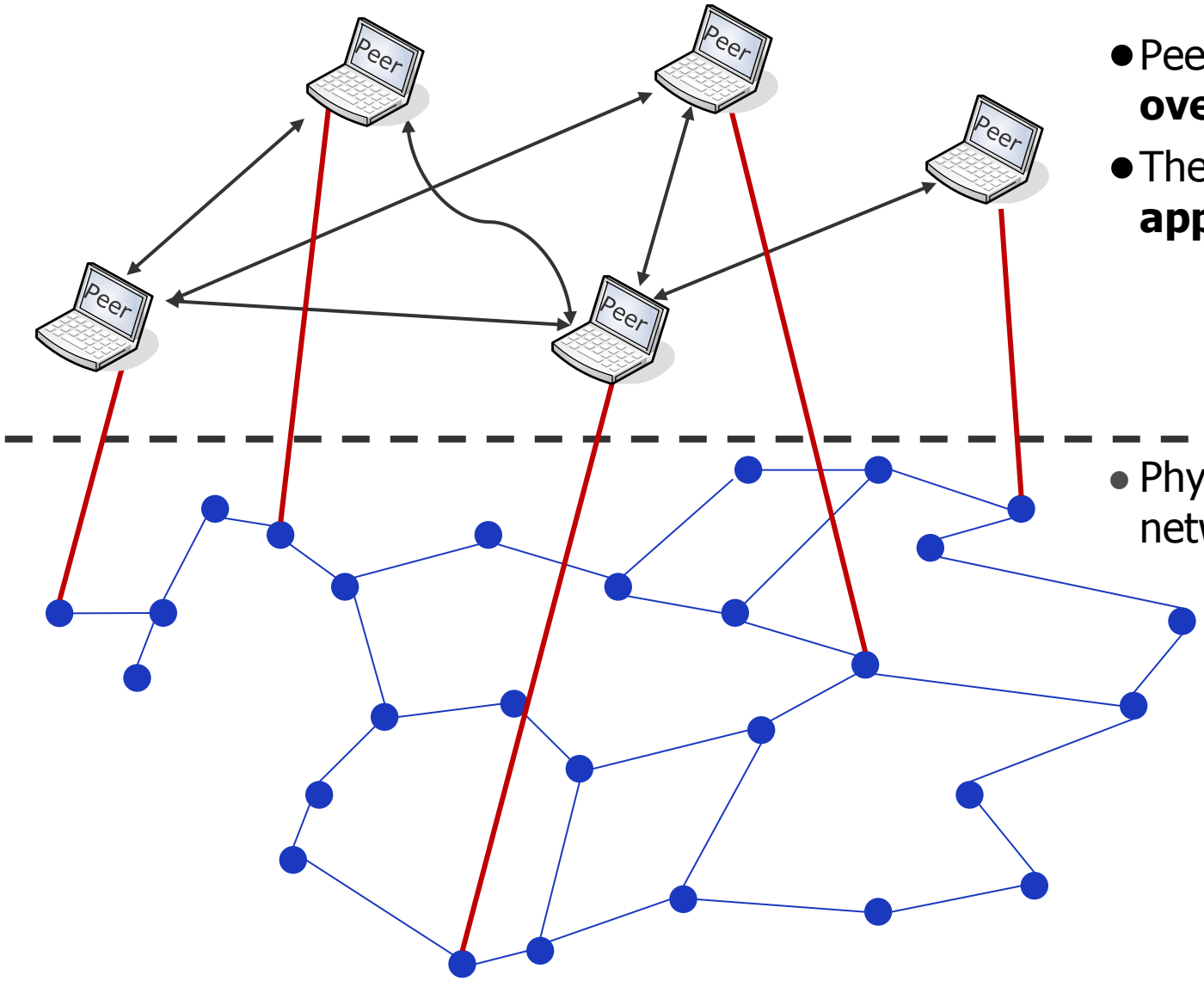
Peer-to-Peer: Introduction

- "Opposite" of Client/Server
- No central server ➡ Information are highly distributed
- Each peer is at the same time **serv**er and cli**ent** (**servent**)
  - Each peer can query, reply to queries, and forward messages at the same time
- Each peer "speaks" directly with other peers





Peer-to-Peer: Concept



- Peer-to-Peer networks are **overlay-networks**
- The focus is on the **application layer**

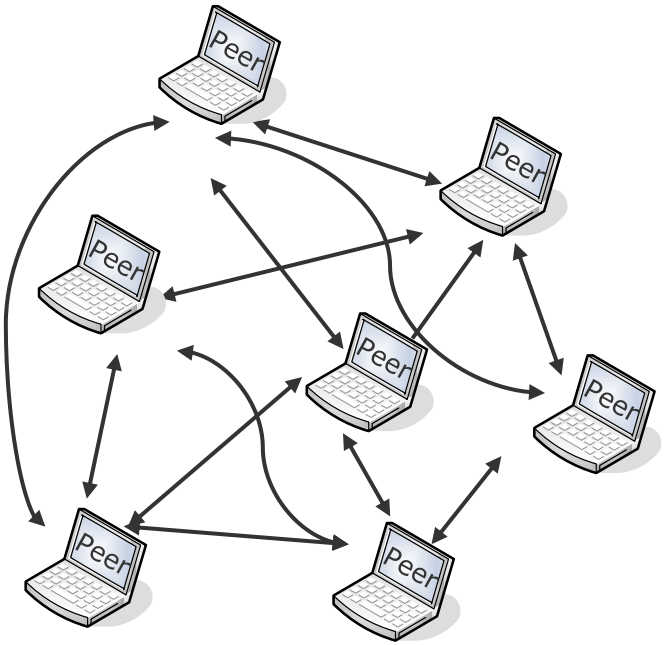
- Physical topology of the network

Peer-to-Peer: Some Applications

- There are many P2P Applications today
  - P2P Telephony, e.g., Skype
  - P2P Instant Messaging, e.g., ICQ
  - P2P Content Sharing, e.g., Gnutella, Kazaa, Freenet, BitTorrent
  - P2P Resource Sharing, e.g., SETI@home
  - P2P Social Networks, e.g., Krawler
  - P2P Video Streaming, e.g., P2PTV, PDTP
  - …

Peer-to-Peer: Properties and Classification

- Peer-to-Peer Networks have some interesting properties
  - No single point of failure
  - Very flexible and adaptive (scalability)
  - Self-organized
  - Reduce/eliminate the demand for server clusters
  - Improve utilization of resources (memory, CPU)
  - Information are always up-to-date
  - Provide better anonymity



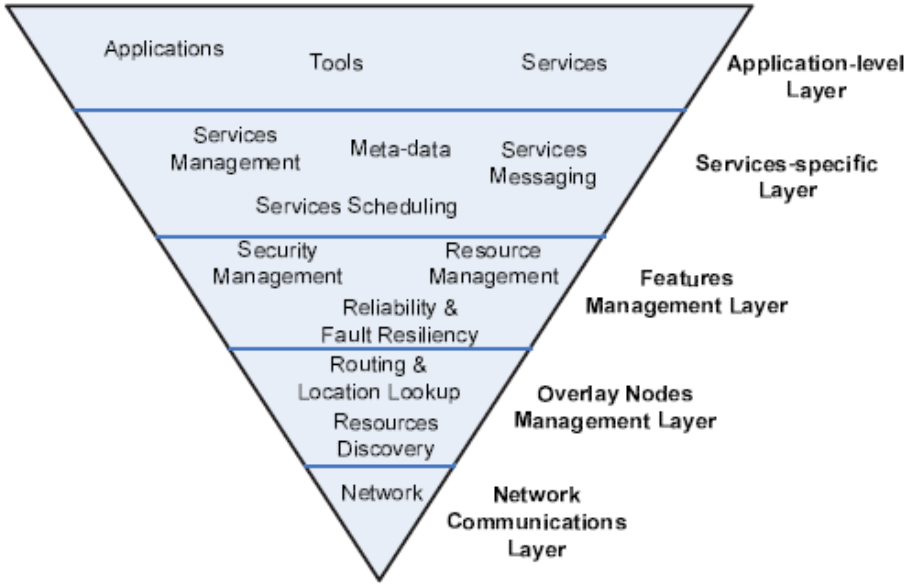
- Classification of Peer-to-Peer Networks
  - Unstructured: Data are placed randomly
  - Structured: Data are placed systematically, e.g., in a Distributed Hash Table (DHT)

Peer-to-Peer: General Problems

- There are also some problems to solve
  - Retrieving of information
  - Discovery of other peers
  - Message routing
  - Keep network traffic low!

Peer-to-Peer: Overlay Architecture

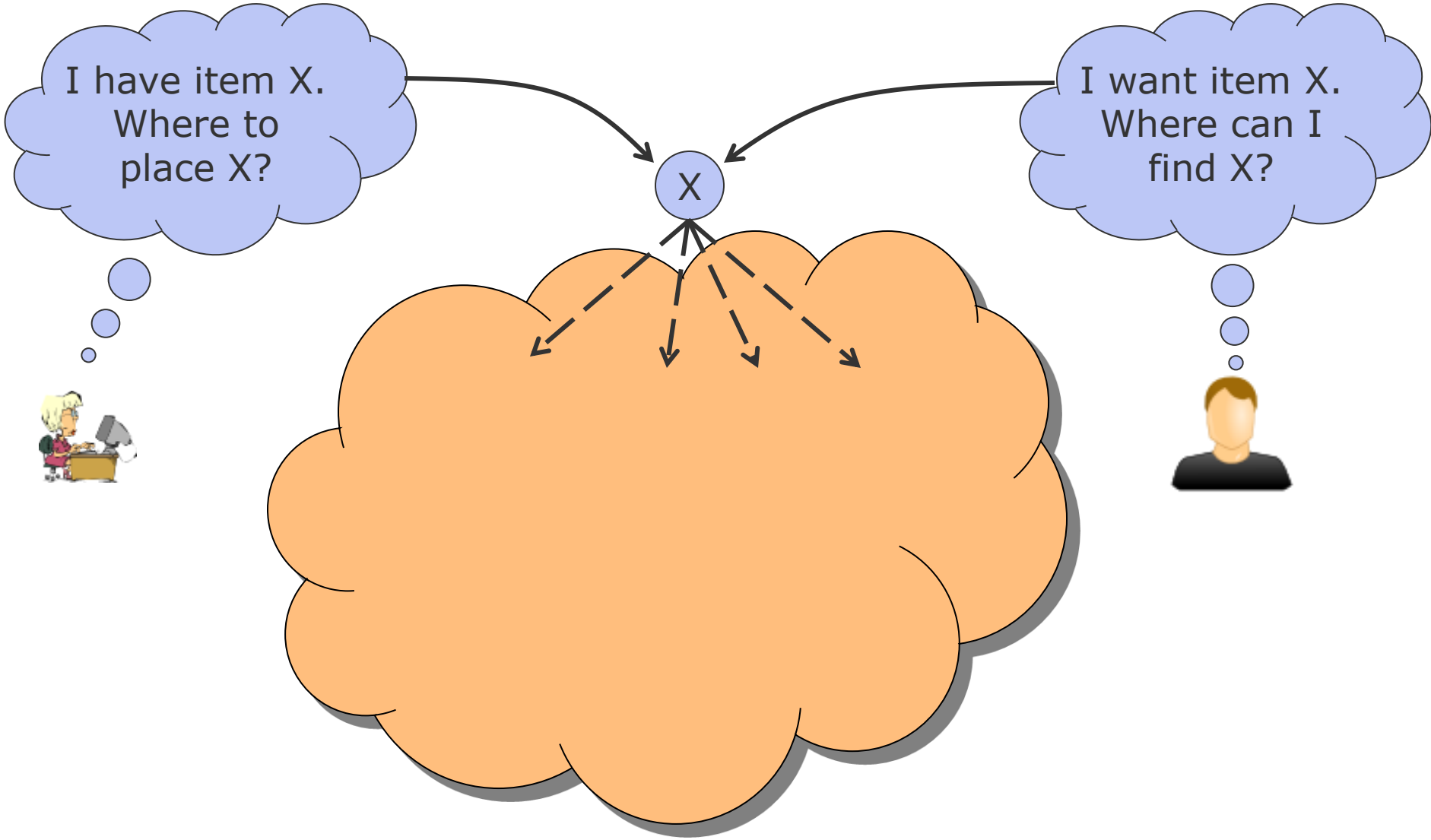
- Overlay architecture
  - Network communications
    - Network characteristics, how is the node connected
  - Overlay node management
    - Discovery of peers, routing
  - Features management
    - Security, reliability, fault resilience
  - Services-specific level
  - Application level
    - Tools on top of the P2P overlay



Main issue: The lookup problem

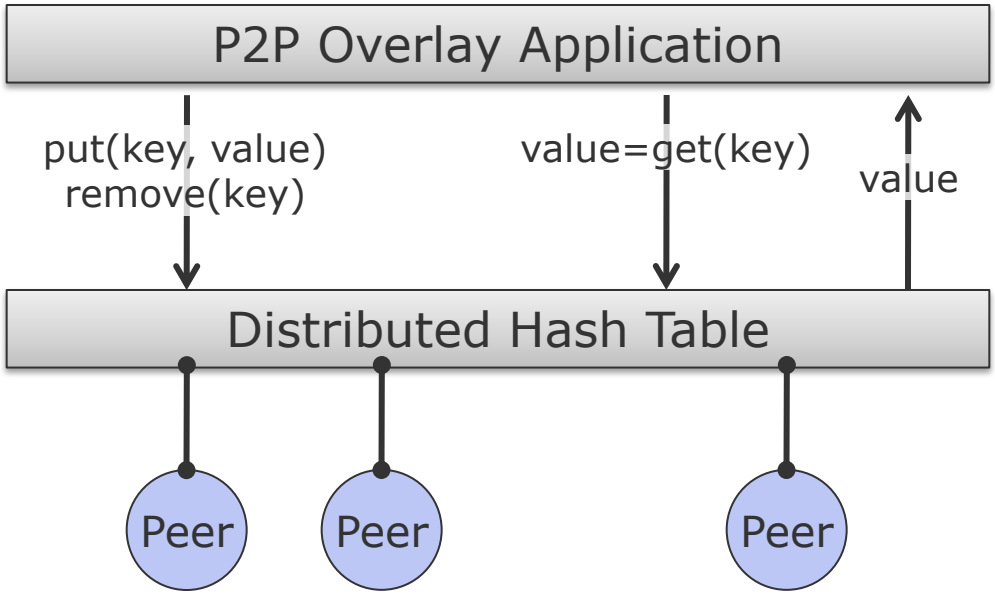
Where to store, and how to find a certain data item in a distributed system without any centralized control or coordination?

Peer-to-Peer: The Lookup Problem



Peer-to-Peer: Overlay Interface to Applications

- Application interface for DHT-based overlay networks



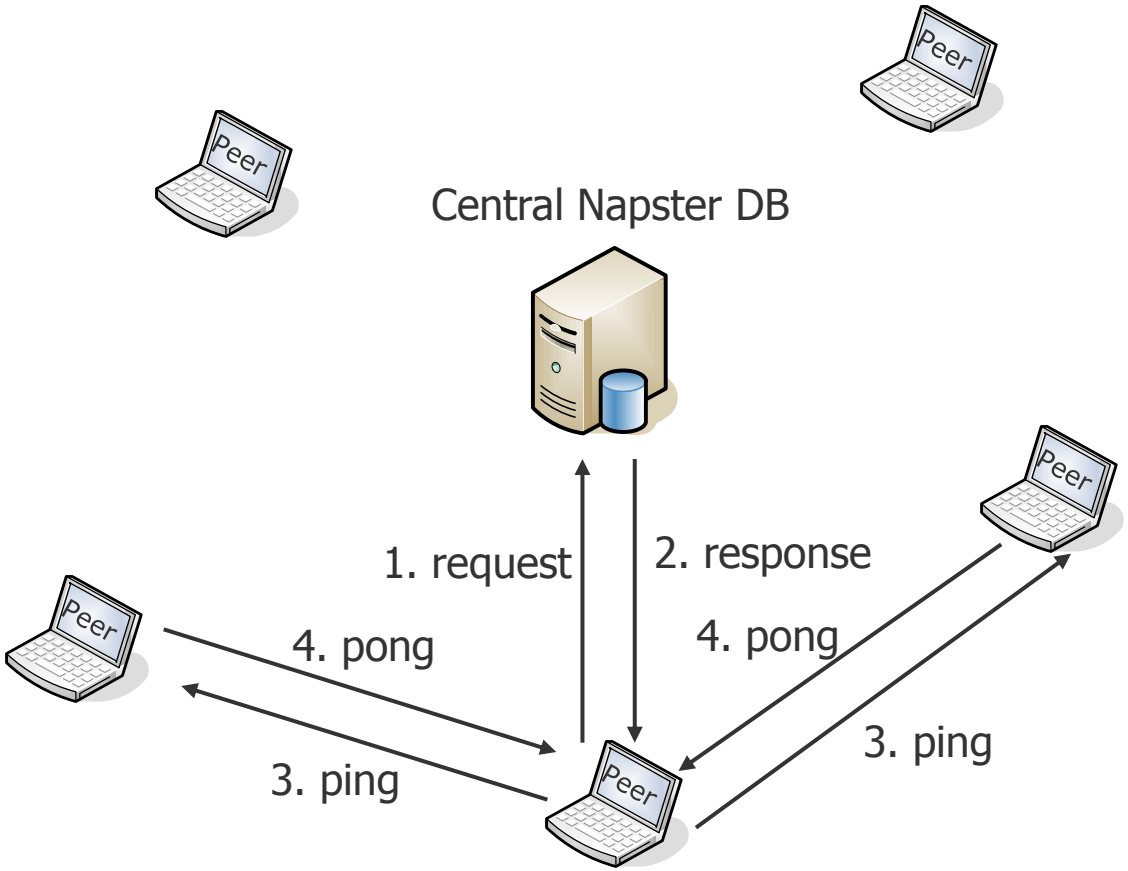
First Generation of P2P-Systems

Napster (1)

- Popular system of the 1. Generation (1999)
- File Sharing as main application
- **Not** a pure Peer-to-Peer network
  - Hybrid system
- Each peer sends the list of shared files to a **central database**
- A **peer asks the central database** when it searches for files
- The central database provides a **list of peers** which store the searched files
- The peer downloads the searched filed **directly** from the peer with the best transfer rate (P2P)

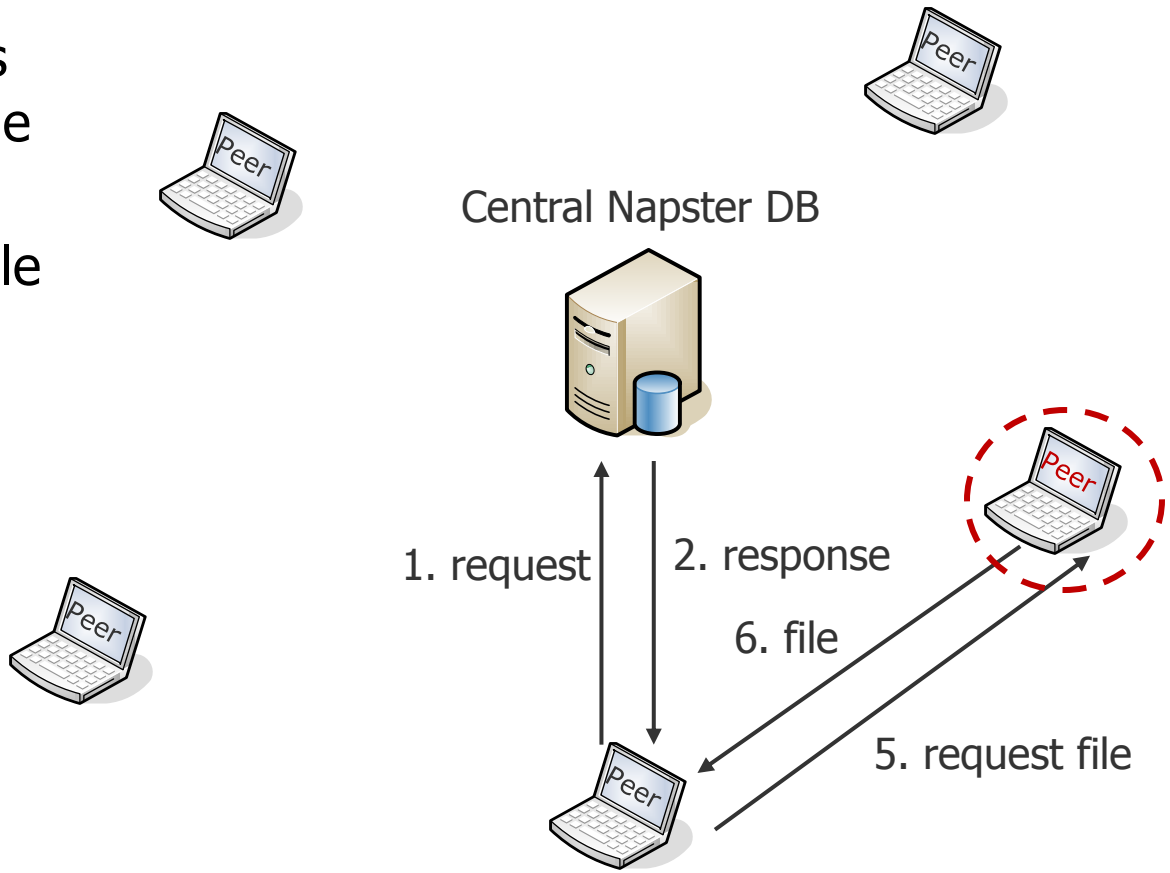
Napster (2)

1. Send request to the central DB
2. PING to all peers provided from the central DB



Napster (3)

1. Send request to the central DB
2. PING to all peers provided from the central DB
3. (Peer-to-Peer) File Download



Napster (4)

- **Advantages**
  - Central database has full knowledge about available items
  - Very fast retrieval of searched information in the central database
  - Supports simple and complicated (fuzy) queries
- **Disadvantage**
  - High traffic load at the central database
  - Available capacity of other peers are not really used
  - Scalability problems
  - Single point of failure!
    - Technical as well as **juridical**

First Generation of P2P-Systems

Gnutella

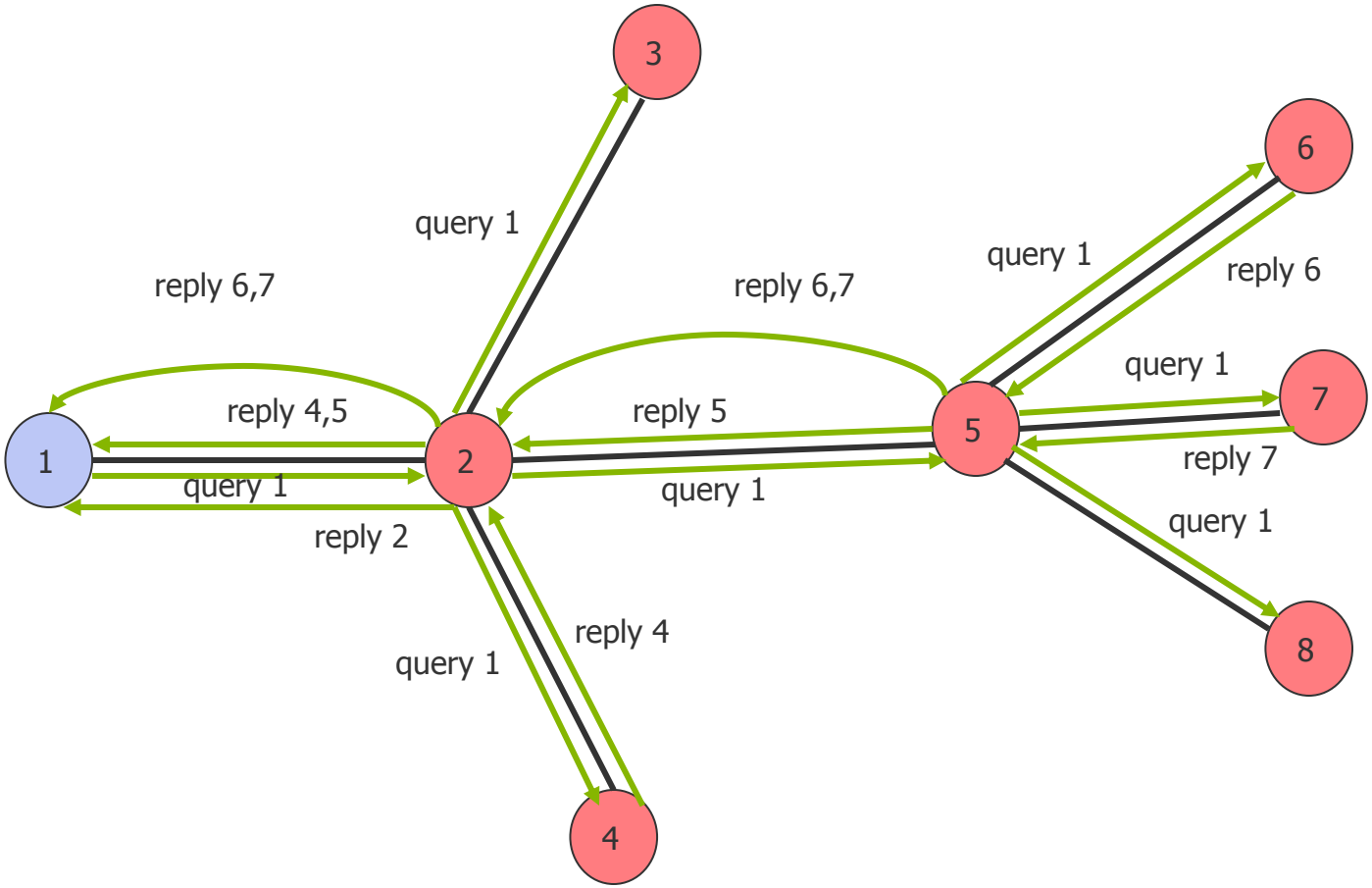
Gnutella: Introduction

- The Gnutella-System
  - Very popular P2P system after Napster
  - **Pure** Peer-to-Peer System
  - No central server
  - Mainly used for file sharing
- Very flexible topology
  - Each node can connect to each other
- Very simple protocol
  - Only 5 different messages
- But
  - No routing-"intelligence" (only Loop-Detection)
  - Messages are always broadcast!!

Messages in Gnutella

Type	Description	Information
Ping	Announce availability and probe for other servents	None
Pong	Response to a Ping	▪ IP address and port number of the responding servent ▪ Number and total kB of files shared
Query	Search request	▪ Minimum network bandwidth of responding servent ▪ Search criteria
QueryHit	Returned by servents that have the requested file	▪ IP address, port number, and network bandwidth of responding servent ▪ Number of results and result set
Push	File download requests for servents behind firewalls	▪ Servent identifier ▪ Index of requested file ▪ IP address and port to send file to

Gnutella: Routing



Gnutella: Generated Traffic in Byte (1)

	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8
N=2	166	332	498	664	830	996	1,162	1,328
N=3	249	747	1,743	3,735	7,719	15,687	31,623	63,495
N=4	332	1,328	4,316	13,28	40,172	120,848	362,876	1,088,960
N=5	415	2,075	8,715	35,275	141,515	566,475	2,266,315	9,065,675
N=6	498	2,988	15,438	77,688	388,938	1,945,188	9,726,438	48,632,688
N=7	581	4,067	24,983	150,479	903,455	5,421,311	32,528,447	195,171,263
N=8	664	5,312	37,848	265,600	1,859,864	13,019,712	91,138,648	637,971,200

Source: "Why Gnutella Can't Scale. No, Really.", Jordan Ritter

- Number of users connected to the Gnutella network: 2000
- Parameters
  - N = Number of connections open
  - T = Number of hops
- Query message length: 83 byte
- Only query relaying (no Responses !!)

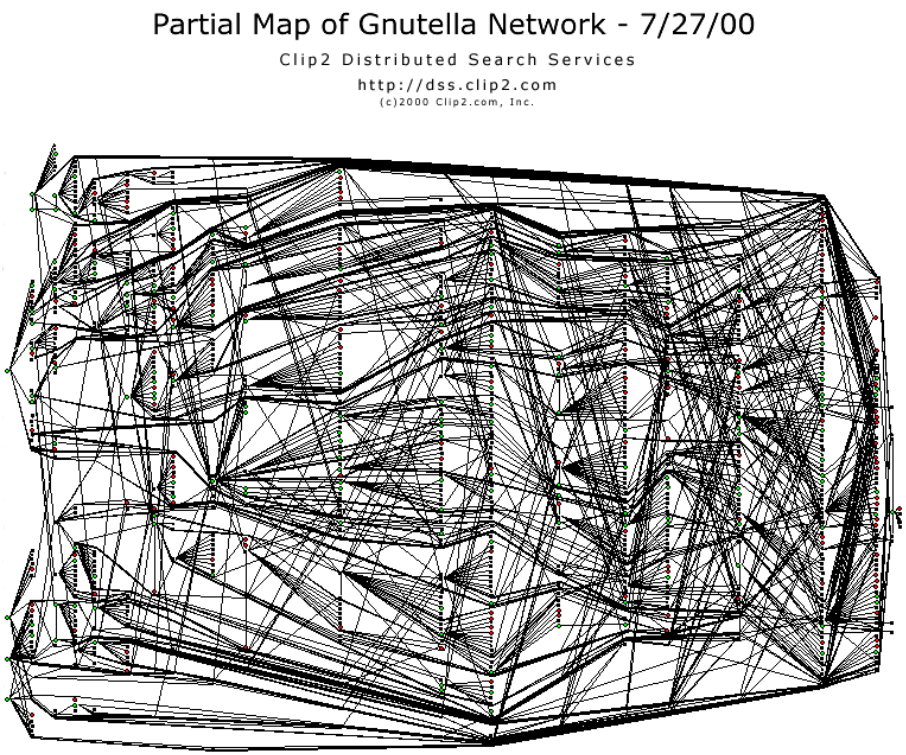
Gnutella: Generated Traffic in Byte (2)

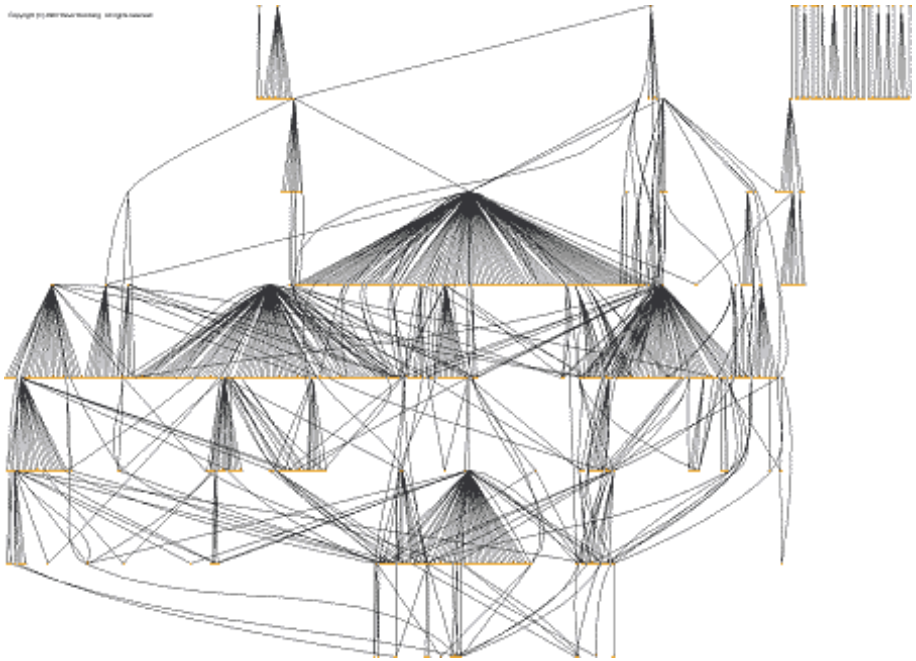
	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8
N=3	283.68	1,418.4	4,822.56	13,900.3	36,594.7	91,061.3	218,15	508.638
N=4	378.24	2,647.68	12,860.2	53,710.1	206,897	758,371	2,688,530	9,306,220
N=5	472.8	4,255.2	26,949.6	147,986	753,17	3,658,050	17,214,200	79,185,000
N=6	567.36	6,240.96	48,793	332,473	2,105,470	12,743,500	74,798,500	429,398,000
N=7	661.92	8,604.96	80,092.3	651,991	4,941,123	35,823,800	252,002,000	1,734,360,000
N=8	756.48	11,347.2	122,55	1,160,440	10,242,000	86,526,900	709,521,000	5,693,470,000

Source: "Why Gnutella Can't Scale. No, Really.", Jordan Ritter

- Mean percentage of users, which share content: **30%**
- Mean percentage of users, which can respond to replies: **40%**
- Mean number of responses per responding user: **10**
- Mean length of responses: **60**
  ➡ "**Standard client settings yield a whopping <u>17MB</u> generated in response to a [...] search query** "

Gnutella: Graphs …





Gnutella: Pros & Cons

- **Advantages**
  - Simple protocol
    ➡ easy to implement
  - Low maintenance traffic
    - Only periodic PING/PONGs
  - No restriction on overlay topology

- **Disadvantages**
  - Message broadcasts lead to network flooding
  - ➡ high network traffic
  - ➡ poor scalability

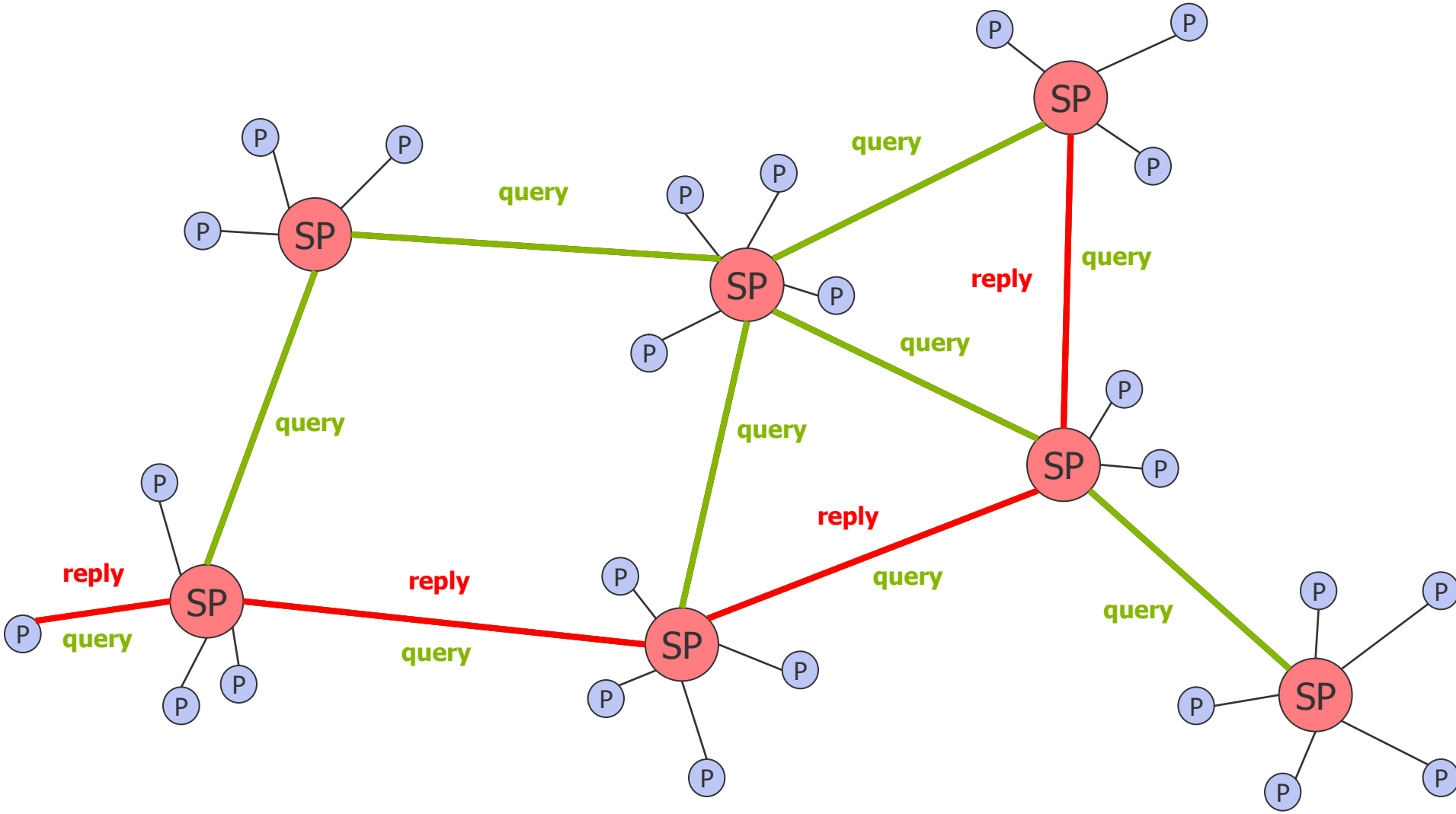
First Generation of P2P-Systems

Super-Peers

Super-Peers

- A possible solution of the scalability problem: **Super-Peers**
  - Concept used in Gnutella 0.6, Kazaa
  - Idea: Nodes with high bandwidth, CPU power, etc. become Super-Peers
    - Super-Peers
    - Normal Peers
  - Only Super-Peers are involved in the **routing process**
  - A "normal" (client) peer connects to a Super-Peer
  - A Super-Peers knows all the contents of its client peers
  - Client peers query their super-peers
  - A Super-Peer may forward a query to other Super-Peers

Super-Peer: Routing



Super-Peers: Pros & Cons

- **Advantages**
  - Reduction of the search process
  - Fast retrieval of data
- **Disadvantages**
  - High traffic at the Super-Peers
  - How to select Super-Peers?
  - What happens when a Super-Peer breaks down?
  - Content of client peers not always possible to index
    - Database contents

Distributed Hash Tables (DHT)

Distributed Hash Tables (DHT): General

- Efficient Look-Up Services required
  - DHTs **guarantee** an **upper bound** for key look-ups
  - DHTs are overlay networks
  - Each **node** and each **object** possesses a **unique ID** (hash key)
  - An **object** is stored on the **node** which is **responsible** for the object ID
  - A node is responsible for an ID if its own ID is the nearest to the object ID

Distributed Hash Tables (DHT): General

- Idea: Overlay topology is not randomly
  - Each node stores in its routing table only the nodes according particular requirements
- All DHTs provide two basic routing operations

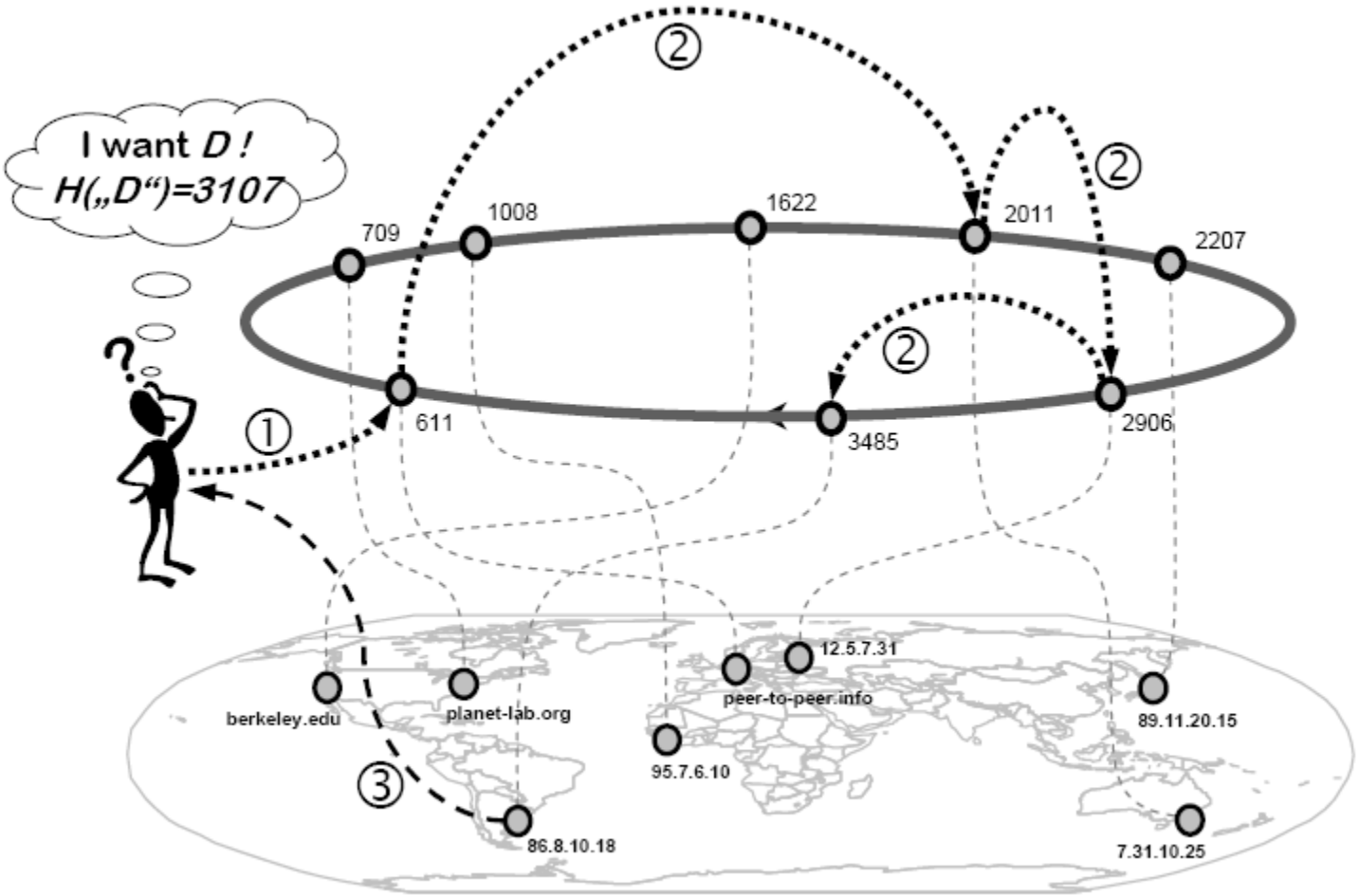
  1. lookup(key) → node
  2. put(object, key) → node

- Popular DHTs:
  - Content Addressable Network (CAN)
  - Chord
  - Pastry
  - Tapestry

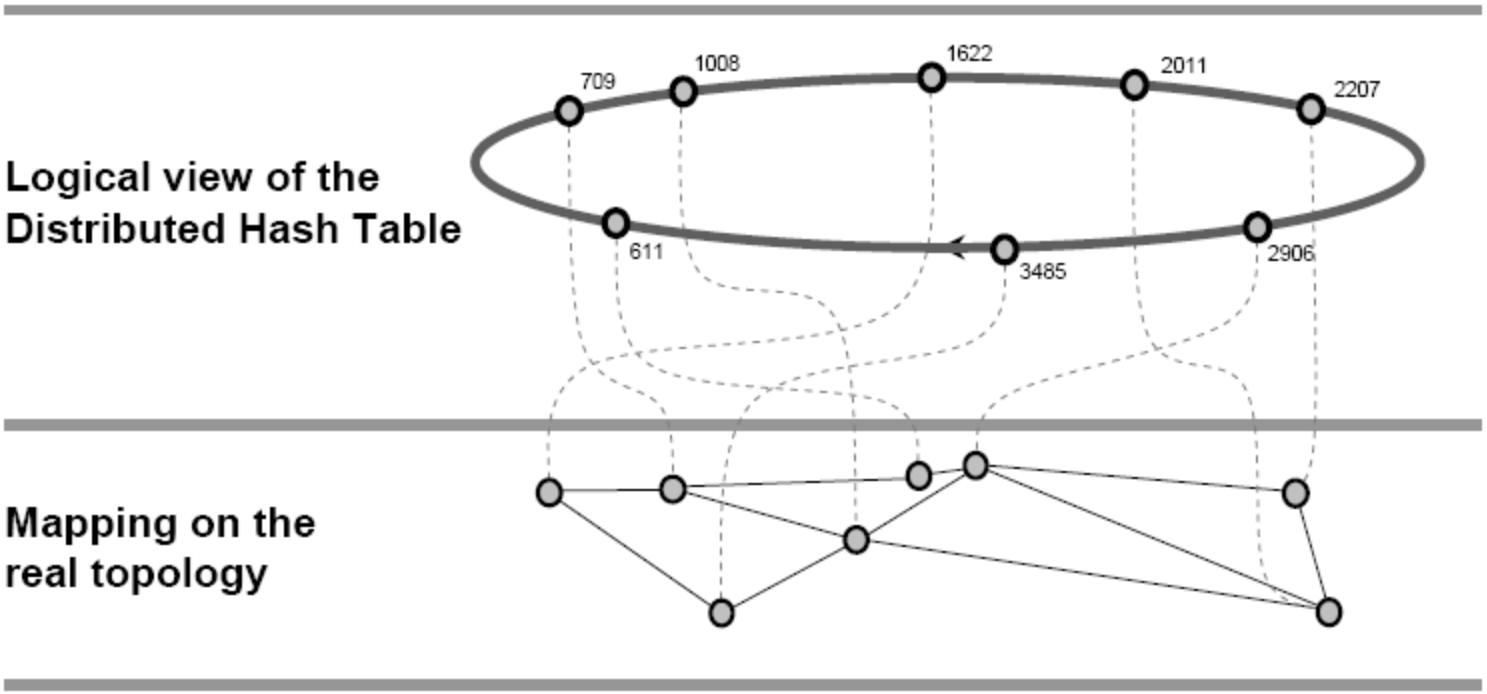
Distributed Hash Tables (DHT): General

- Each node manages a small number of references to other nodes.
  - O(log N) references, N number of nodes
- By mapping nodes and data items into a common address space, routing to a node leads to the data items for which a certain node is responsible
- Queries are routed via a small number of nodes to the target node
  - An item can be located by routing via O(log N) hops.
- Identifiers are distributed on the nodes nearly equally
  - Load for retrieving items may be balanced among the nodes
- No node plays a particular role
  - Formation of hot spots or bottlenecks can be avoided.
  - Departure of a node does not have considerable effects on the DHT
  - DHTs are considered robust against random failures and attacks
- Distributed index provides a definitive answer about results
  - If a data item is stored in the system, the DHT guarantees that the data is found

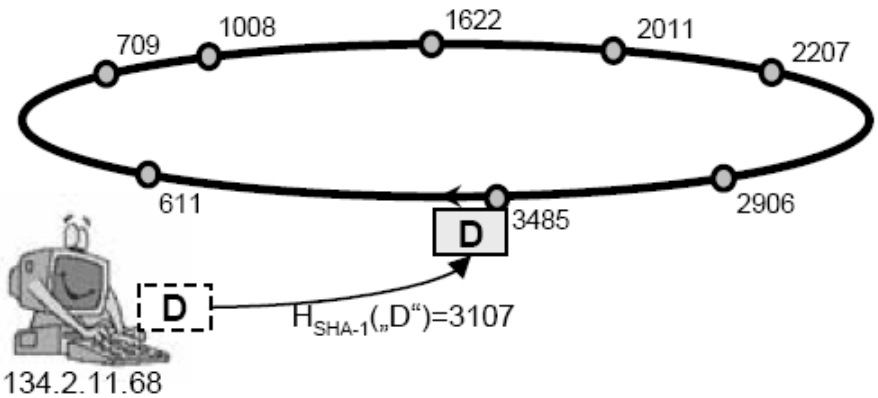
Distributed Hash Tables (DHT): General



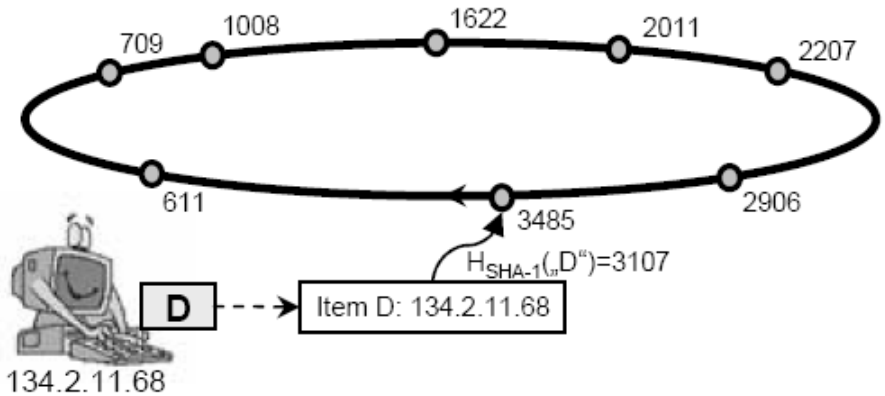
Distributed Hash Tables (DHT): General



Distributed Hash Tables (DHT): General



(a) Direct Storage



(b) Indirect Storage

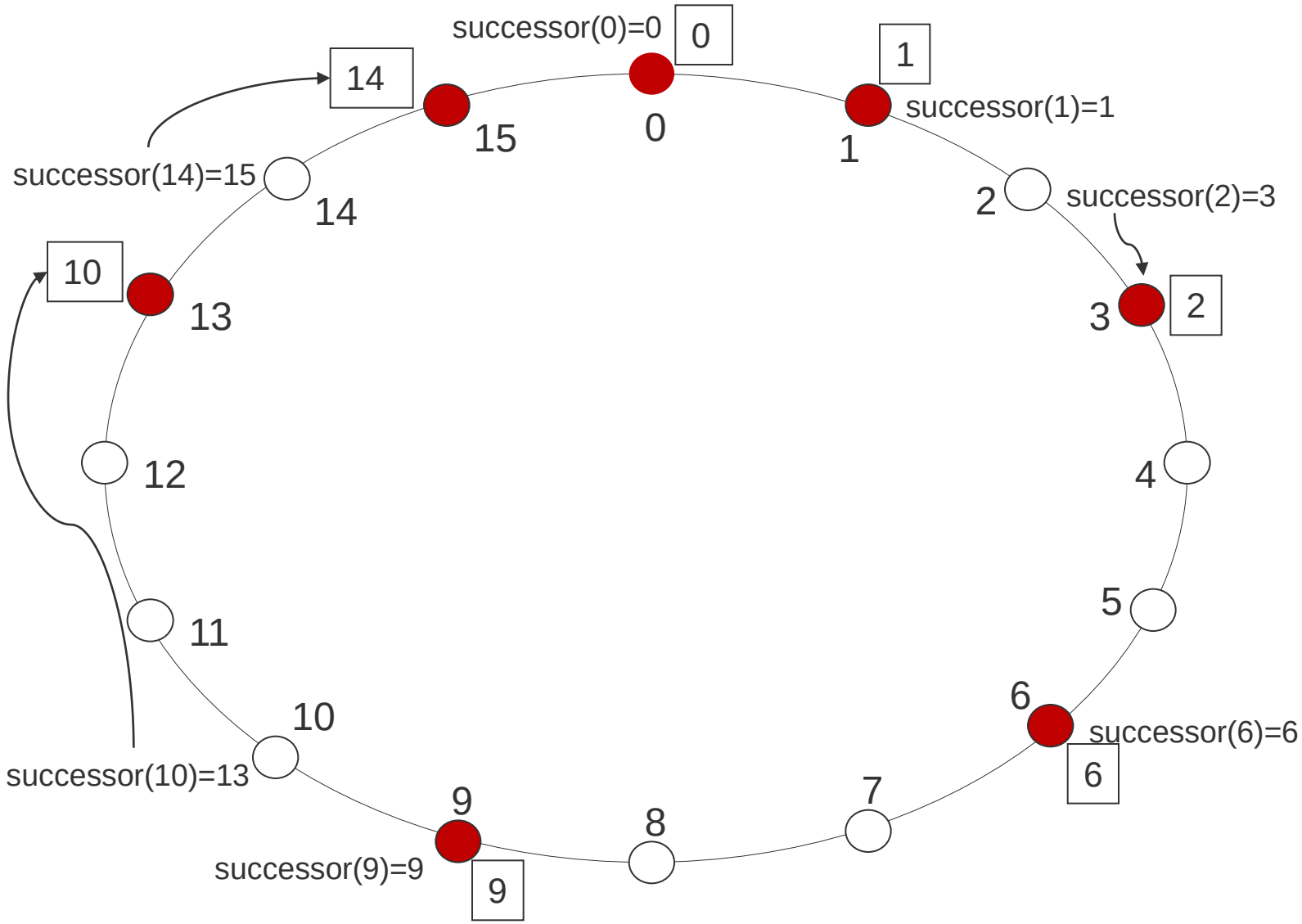
Distributed Hash Tables (DHT)

Chord

Case Study I: Chord (1)

- The IDs are arranged in a **virtual** ring from 0 – (2^m -1)
- Consistent hashing is used
  - The addition and removal of slots into the hash does not significantly change the mapping of the keys
- Consistent Hashing (e.g. SHA-1) assigns each **node** (e.g. peer) **and** each **object** a m-bit ID in the same overlay ID ring
  - For nodes the IP address is hashed
  - For data the data key (e.g. file name) is hashed
- New nodes take their position in the ring according their ID
- The key k is assigned to the first node with ID ≥ k
  ➡successor(k)
- Chord provides an upper bound of $O(\log N)$ for the key look-up
  - N is the number of peers in the system

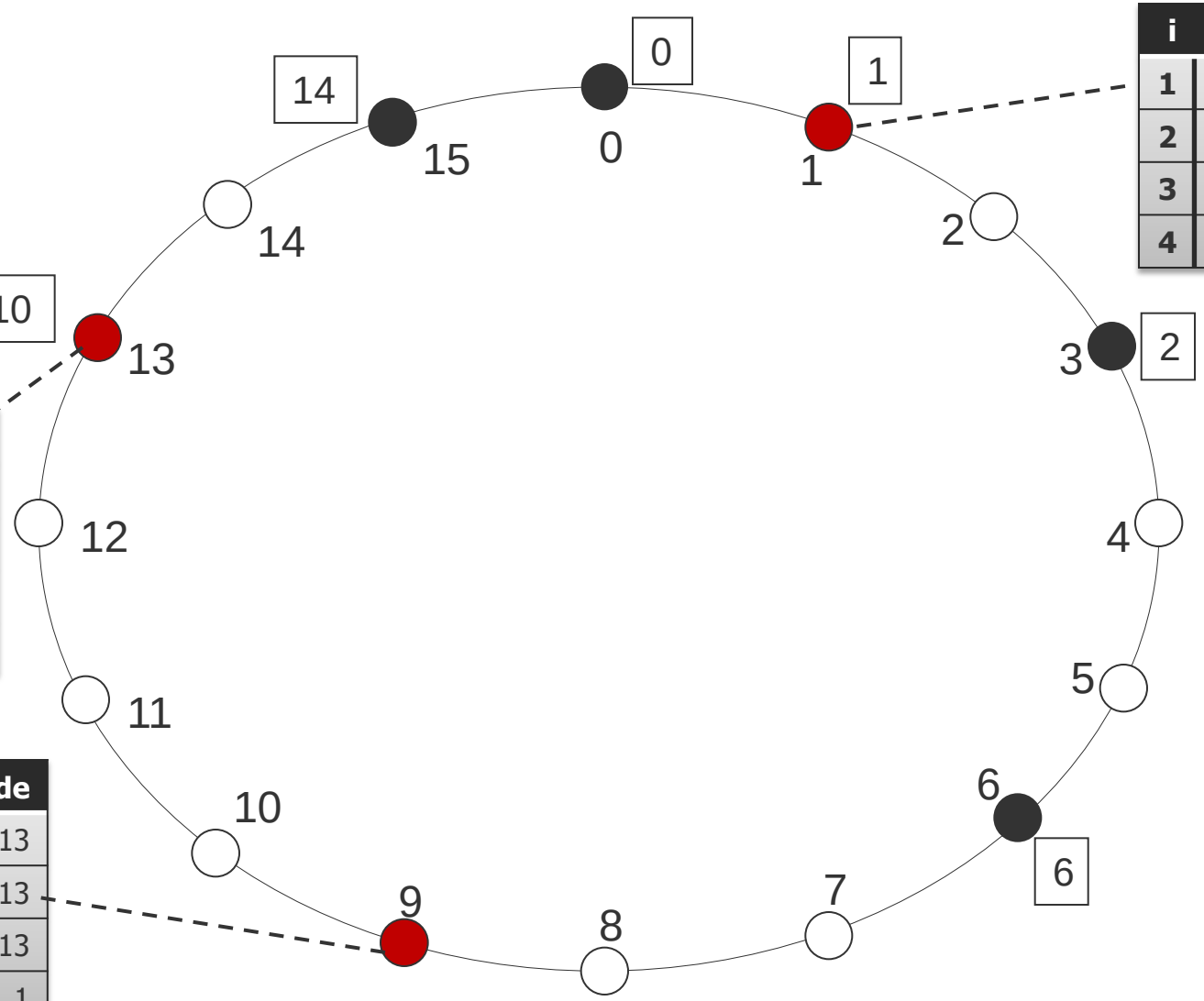
Case Study I: Chord (2)



Case Study I: Chord (3)

- Each node n has a routing table with max. m entries
  - Finger table
- The i-th entry = first node s with an ID at least 2^{i-1} larger than n's ID
  - s = successor(node.ID + 2^{i-1})
- Peer s is the i-th finger of peer node

Case Study I: Chord (4)

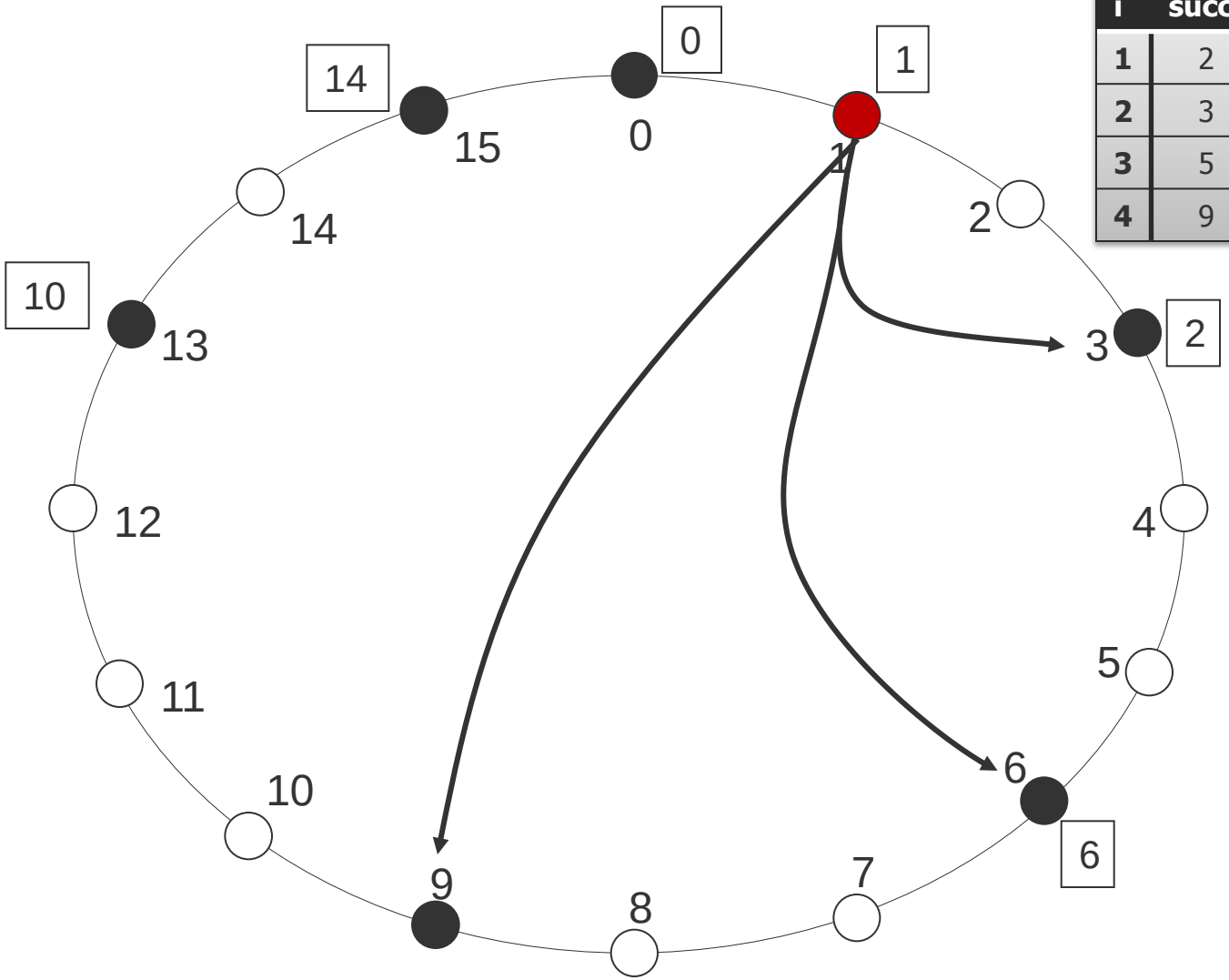


i	succ	node
1	2	3
2	3	3
3	5	6
4	9	9

i	succ	node
1	14	15
2	15	15
3	1	1
4	5	6

i	succ	node
1	10	13
2	11	13
3	13	13
4	1	1

Case Study I: Chord (5)



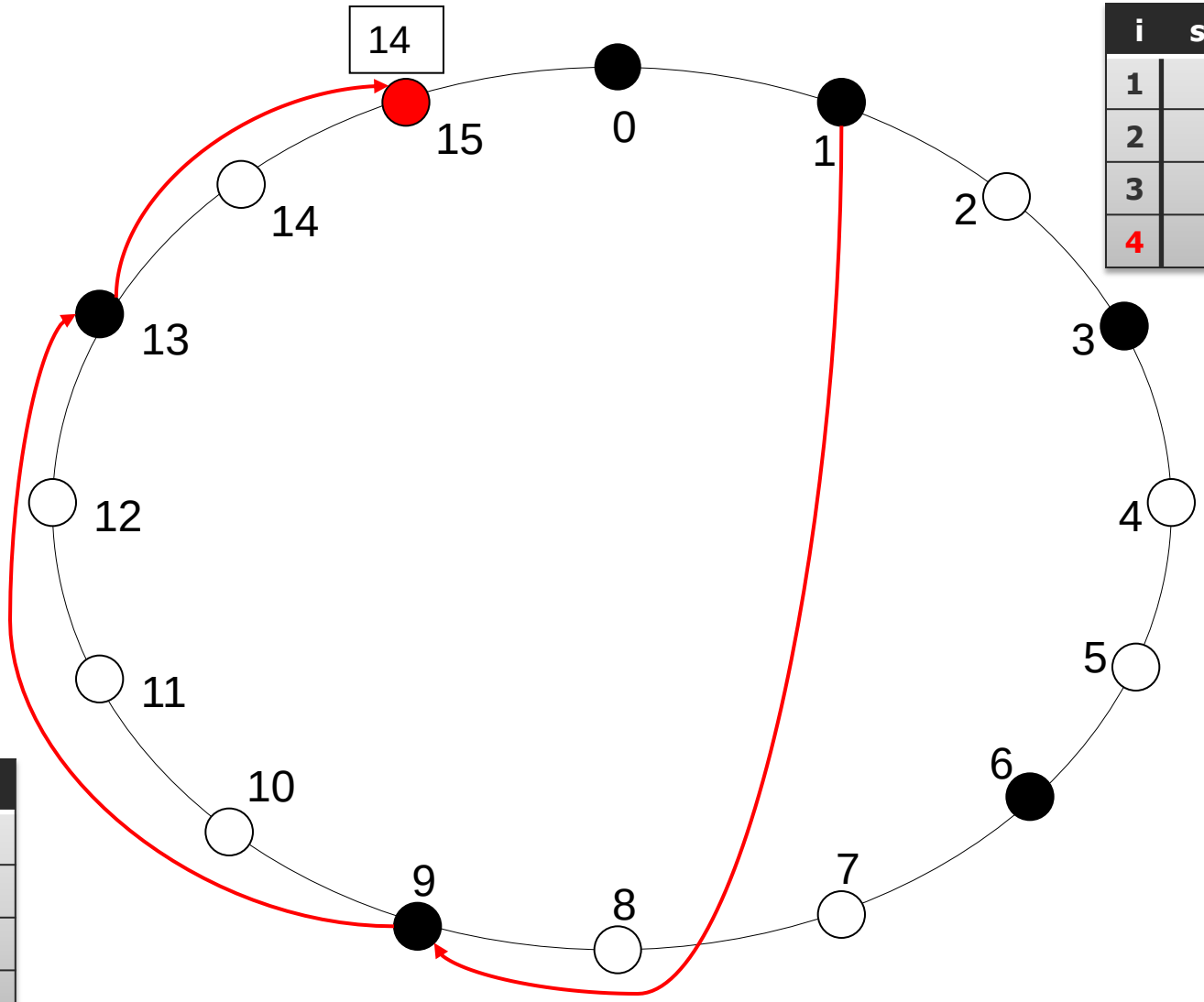
i	succ	node
1	2	3
2	3	3
3	5	6
4	9	9

Case Study I: Chord – Routing (1)

- **Routing:** if node n searches for the key k
  - If successor(k) unknown, forward query to next node n' in the finger table
  - If n' does not know the successor(k), n' forwards the query to next node in its finger table
  - ➡ This is done until the successor(k) is found

  - **Idea**: in each step, the distance to the successor(k) is halved
    - ➡ $O(\log N)$ many (overlay-) hops
    - ➡ Chord finds a key in $O(\log N)$

Case Study I: Chord – Routing (2)



i	succ	node
1	2	3
2	3	3
3	5	6
4	9	9

i	succ	node
1	14	15
2	15	15
3	1	1
4	5	6

i	succ	node
1	10	13
2	11	13
3	13	13
4	1	1

Case Study I: Chord – Stabilize

- Each node n performs a stabilize-procedure to correct possible network anomalies
  - Node n asks its successor for its predecessor p
  - Node n tests, whether p has to be its successor, i.e., p is between n its successor
- Node n updates periodically one finger f which is selected randomly.
  - For this node n searches for successor(node.ID + 2^{x-1}) and updates it with the new successor(node.ID + 2^{x-1})

Chord-based Applications

- On top of Chord many applications can be realized
  - Cooperative File System (CFS)
    - Multiple providers of content cooperate to store data
    - The total load is spread over all participants
  - Chord-based DNS
    - Distributed DNS lookup service
    - Host names are keys and IP address are values

Distributed Hash Tables (DHT)

Pastry

Case Study II: Pastry – Introduction

- Similar concept as Chord
  - Nodes are organized in a virtual ID ring from 0 to $2^{128} - 1$
  - Each node and object is assigned a 128 bit key (hash key)
  - Each ID is sequence of numbers in the base 2^b (b=1 binary, b=4 hex)
- But:
  - The routing is based on prefix matching
  - Considers the topological distance
  - Each node maintains
    - a routing table
    - a leaf set

Case Study II: Pastry – Routing Table (1)

- Routing table
  - $\log_{2^b} N$ rows
  - Each row has $2^b - 1$ entries
  - Each entry in row n has the same n digit prefix as the current node
  - Each entry in row n differs on the (n+1)-th position from the current node
  - Each of these entries has one of the 2^b -1 digits on the (n+1)-th position

Case Study II: Pastry – Routing-Table (2)

- Routing table
- NodeID **34035761**, b = 3

row	0	1	2	3	4	5	6	7
0	03761261	13541241	27235106		45521251	53610176	65547772	71154601
1	30771255	31521225	32236556	33400741		35716236	36124163	37715026
2		34102724	34203162	34303735	34404216	34576555	34653437	34734152
3	34006624	34012773	34025354		34045631	34050032	34064645	34076214
4	34030342	34031643	34032346	34033652	34034040		34036123	34037437
5	34035073	34035152	34035031		34035070		34035667	
6	34035701	34035717			34035740			34035774

Case Study II: Pastry – Neighborhood Set & Leaf Set

- Leaf set
  - Contains the L closest nodes
  - $L/2$ smaller nodes
  - $L/2$ larger nodes
- Typical values for $|L|$ are 2^b or $2 * 2^b$

Case Study II: Pastry – Node State

Routing Table

Local Overlay ID: 34035761

0	1	2	3	4	5	6	7
03761261	13541241	27235106		45521251	53610176	65547772	71154601
30771255	31521225	32236556	33400741		35716236	36124163	37715026
	34102724	34203162	34303735	34404216	34576555	34653437	34734152
34006624	34012773	34025354		34045631	34050032	34064645	34076214
34030342	34031643	34032346	34033652	34034040		34036123	34037437
34035073	34035152	34035031		34035070		34035667	
34035701	34035717			34035740			34035774

Leaf Set

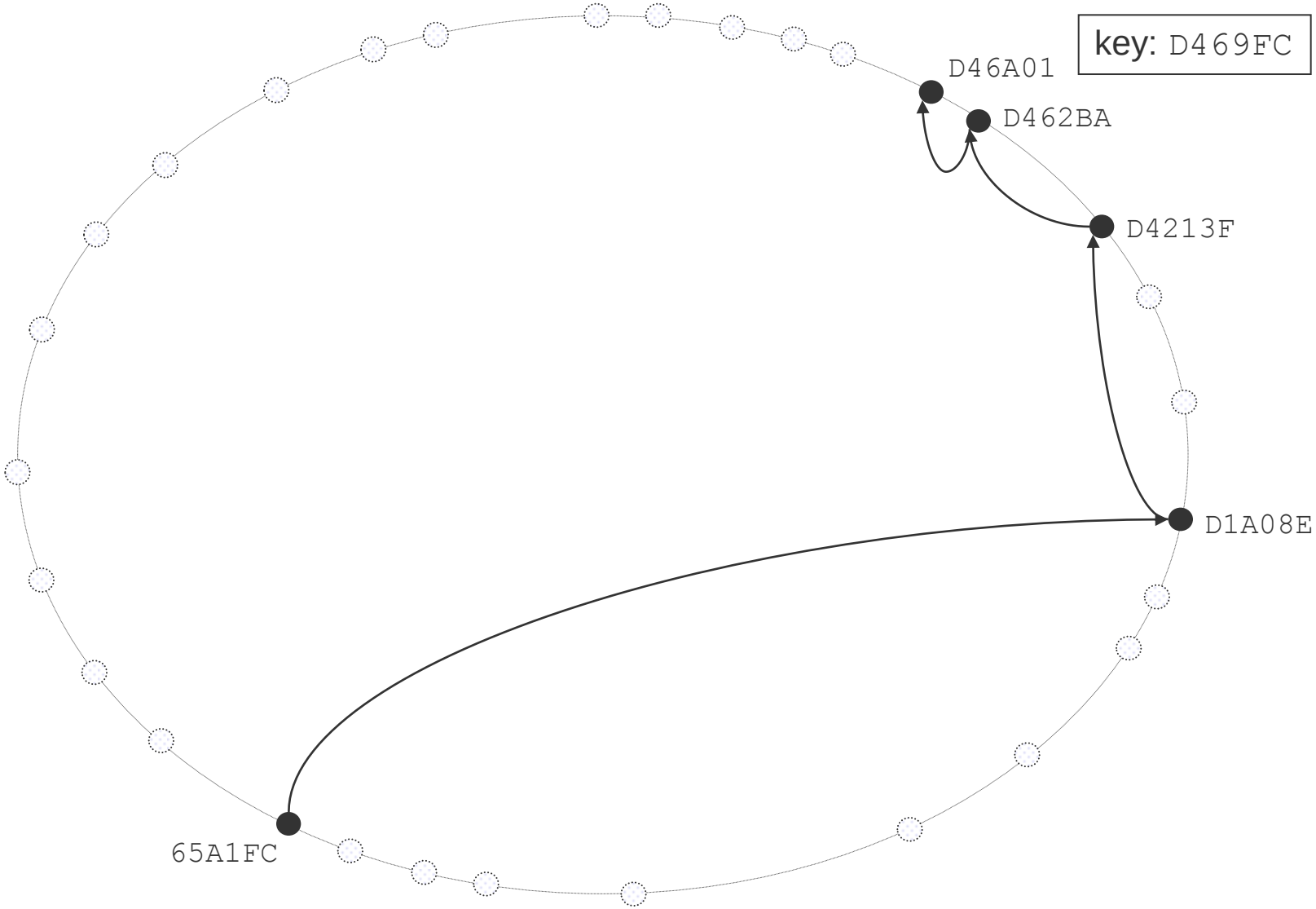
smaller	34035667	34035701	34035717	34035740
larger	34035774	34036012	34036042	34036076

Case Study II: Pastry – Routing (1)

- Routing in Pastry
  - Node n searches the leaf set for nodes which are responsible for key k
  - If the leaf node l is responsible for key k, the query is addressed directly to node l
  - Otherwise the routing table is searched for a node with larger prefix matching (at least one more position) as node n
  - If such a node does not exists the leaf set is searched for a node which is closer to k

- This process is repeated in each node until the query arrives at the node which is responsible for key k
  - Intuitive: with each routing step the matching prefix grows for one position
  - ➡ $O(log_2 b\ N)$ (Overlay-) routing effort

Case Study II: Pastry – Routing (2)



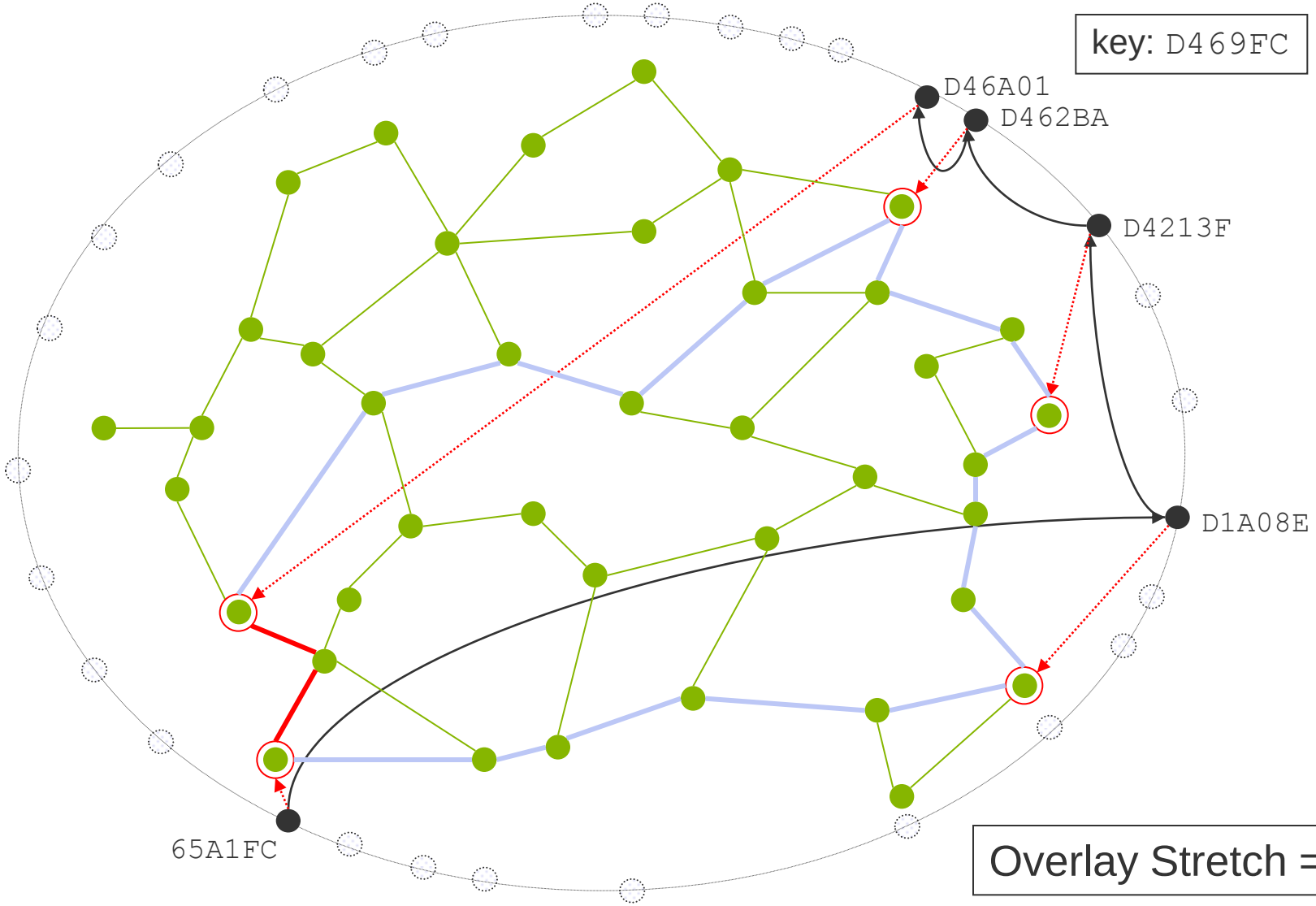
Overlay- vs. Physical Topology

Overlay vs Physical Topology (1)

- DHTs can route very efficient in the overlay network
- However, there is no relationship between the overlay- and physical topology
  - Neighbors in the overlay network are rarely neighbors on the physical network
  - DHTs may show high **Overlay Stretch**
    - The ratio of the accumulated physical route during the overlay routing for the search of a key and the real path length between source and destination nodes

$$\text{Overlay Stretch} = \frac{\text{Real path length}}{\text{Path length in overlay network}}$$

Overlay vs Physical Topology (2)



Overlay Stretch = 17/2

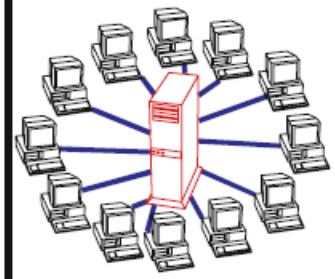
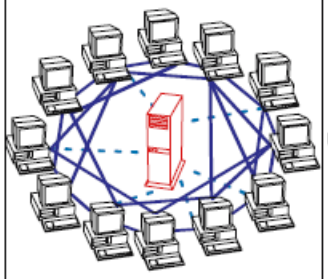
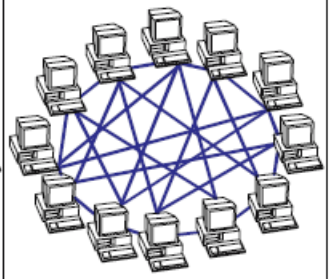
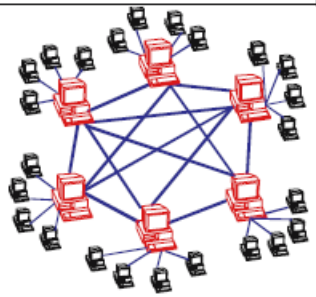
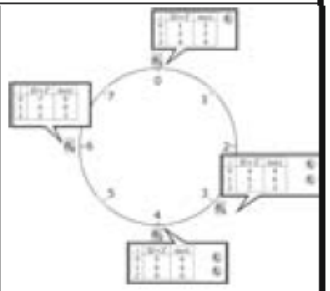
Overlay vs Physical Topology (3)

- Several methods to reduce the overlay stretch
  - **Proximity Neighbor Selection (PNS)** in Pastry
  - Each Pastry-node selects periodically a random entry from row i from its routing table
  - This random node replies with the i-th row from its routing table
  - After the response, the node computes pair wise the distance to the own entries and the entry from the response (e.g. PING/PONG)
  - If needed the own entry is replaced by the entry from the response if it is physically closer than the own entry (e.g. hop count, round trip time)

Overlay vs. Physical Topology (4)

- **Landmarking**
  - Fixed set of Landmark-Nodes
  - Each node measures periodically its distance to all Landmark-Nodes
    - ➡ Sequence of the Landmark-Nodes
  - **Idea**: Nodes with the same Landmark-Distance-Sequence are with high probability physically close to each other
  - ➡ Nodes with same Landmark-Distance-Sequence assign each other numerical close overlay Ids
  - Example: 16 Landmark-Nodes in a Pastry-Network with hex IDs
  - Nodes with the same Landmark-Distance-Sequence assign to each other the same overlay-ID-prefix

Summary

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
				

Summary

- The Peer-to-Peer principle is not really new
- However, with the growth of the Internet and emerging of new applications a new wave of P2P applications was created
- A P2P network is a overlay network on the Application Layer
- In a P2P network all peers have the same role
  - Server + Client = Servent