Netzwerktechnik und IT-Netze

Chapter 6: P2P

Vorlesung im WS 2016/2017

Bachelor Informatik

(3. Semester)

Prof. Dr. rer. nat. Andreas Berl

Fakultät für Elektrotechnik, Medientechnik und Informatik

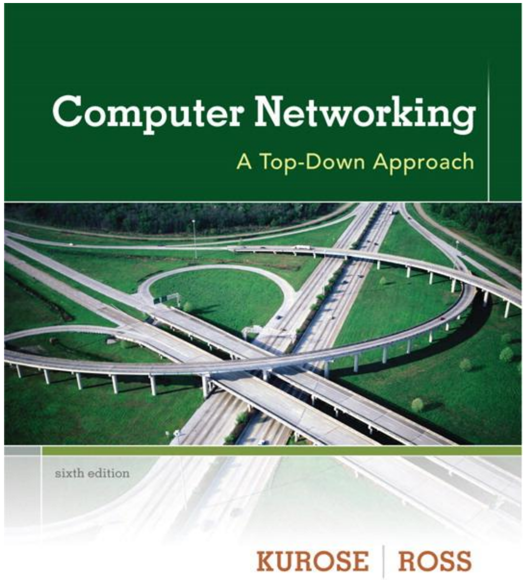
Overview

- Introduction
- Computer Networks and the Internet
- Application Layer
  - WWW, Email, DNS, and more
  - Socket programming
- Transport Layer
- Network Layer
- Link Layer
- P2P Networks
- Firewalls

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

Literatur und Quellen

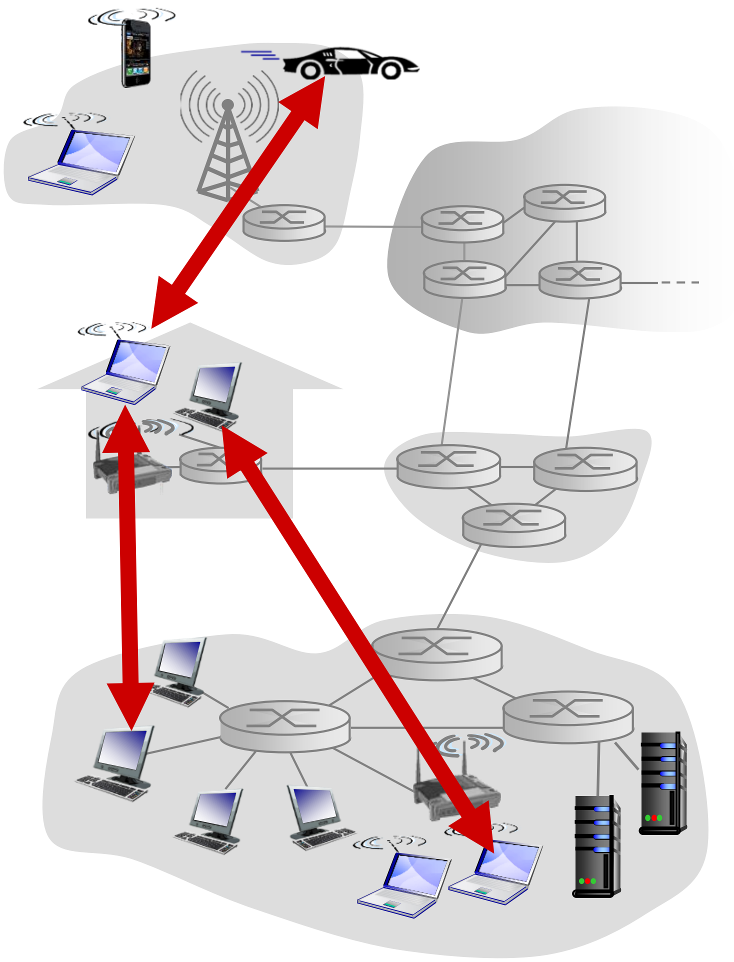
- A note on the use of these power point slides:
  - All material copyright 1996-2012© J.F Kurose and K.W. Ross, All Rights Reserved
  - Do not copy or distribute this slide set!



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Example: Pure P2P architecture

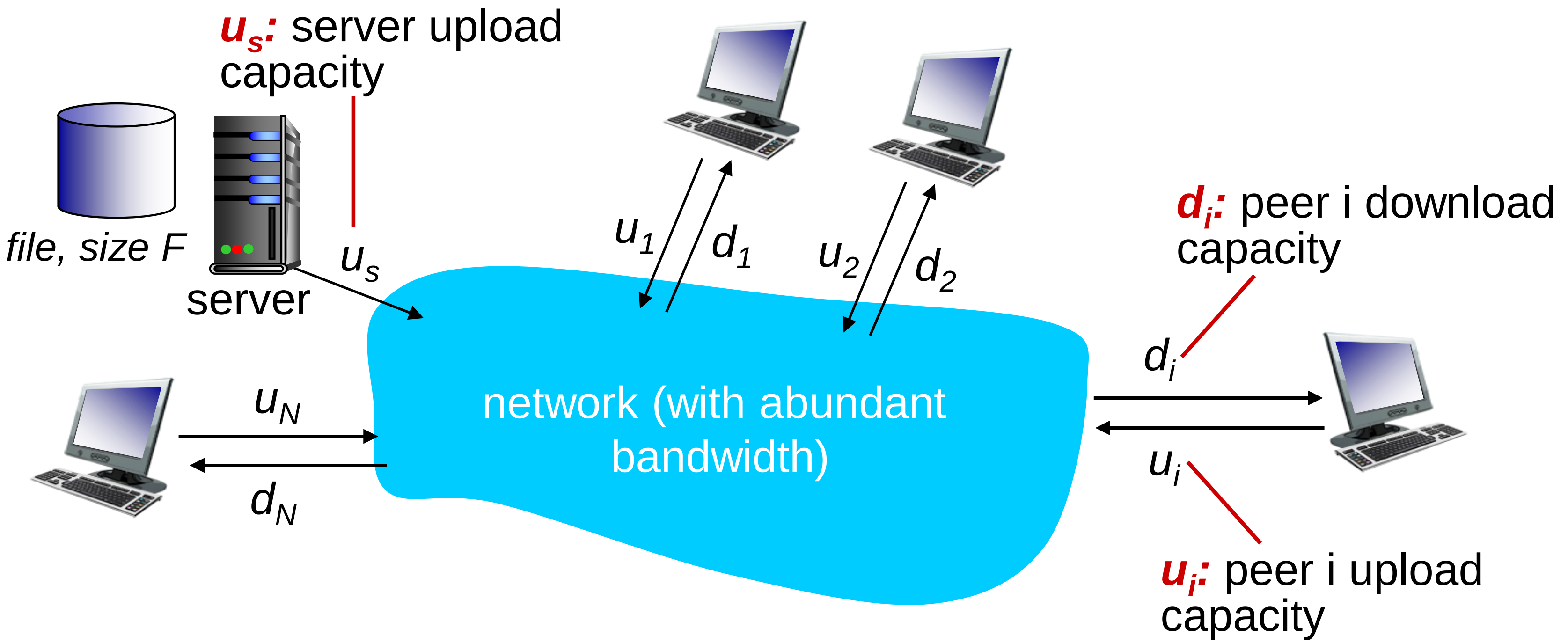
- No always-on server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change IP addresses
- Examples:
  - File distribution (BitTorrent)
  - Streaming (Kankan)
  - Voip (Skype)



Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

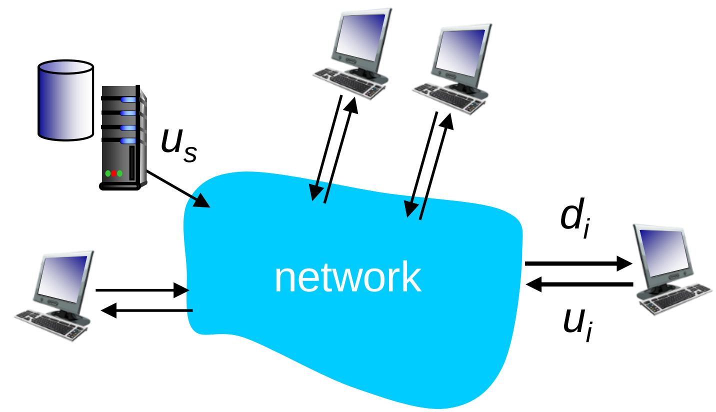
File distribution: Client-Server vs. P2P

- Question: How much time to distribute file (size F) from one server to N peers?
  - Peer upload/download capacity is limited resource



File distribution time: Client-server

- Server transmission: Must upload N file copies in parallel:
  - Time to send one copy: F/u_s
  - Time to send N copies: NF/u_s
- Client: Each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}



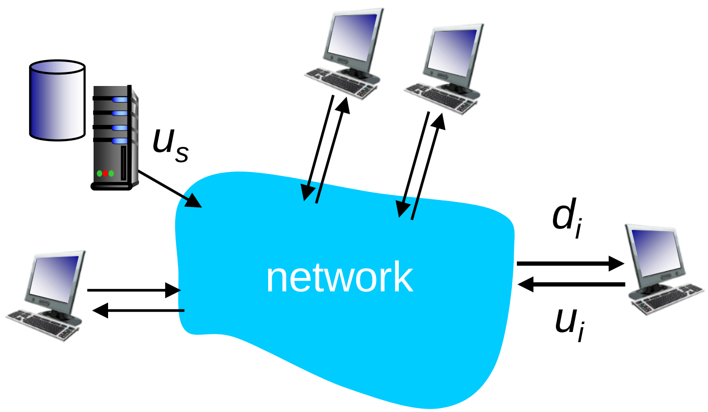
Time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} > max\{NF/u_s, F/d_{min}\}$$

Increases linearly in N

File distribution time: P2P

- Server transmission: Must upload at least one copy
  - Time to send one copy: F/u_s
- Client: Each client must download file copy
  - Min client download time: F/d_{min}



- Clients: As aggregate must download NF bits
  - Max upload rate (limiting max Download rate) is $u_s + \sum u_i$

Time to distribute F to N clients using client-server approach

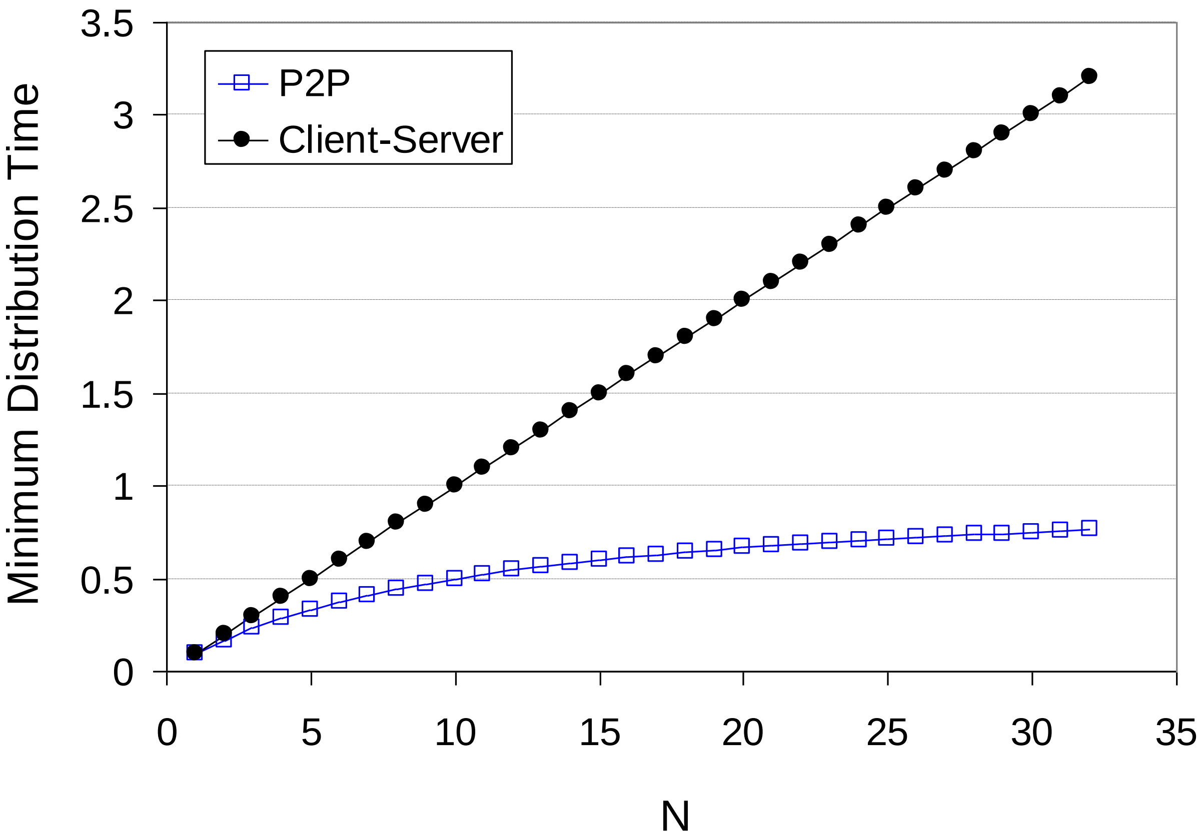
$$D_{P2P} > max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$

Increases linearly in N

… but so does this, as each peer brings service capacity

Client-server vs. P2P: Example

- Client upload rate = u, F/u = 1 hour, u_s = 10u, $d_{min} \geq u_s$



The Architecture of P2P Systems

- Layering of a P2P system

User

Application Layer

Realization of specific applications:
stock quotes, data base, messaging etc.

Overlay Layer

De-centralization: abstraction of
distributed information management
and distributed resources

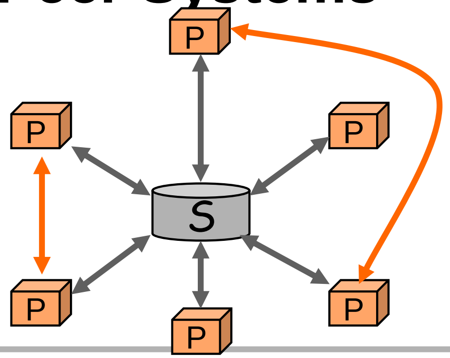
Network Layer

„Real“ transport network:
e.g., Internet, telephone network, …

P2P layers

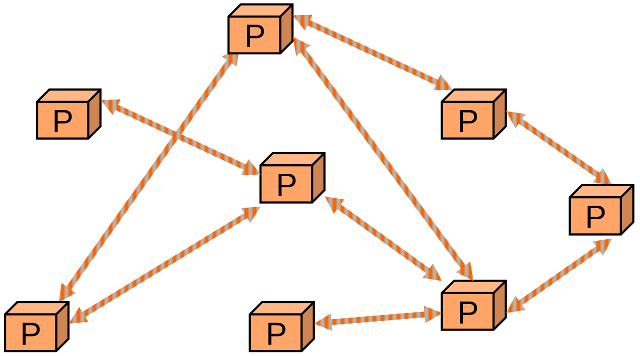
2.3 A Classification of Peer-to-Peer Systems

- Centralized P2P systems
  - Centralized Server connects Peers
  - Direct interaction between peers
  - Joint usage of distributed resources



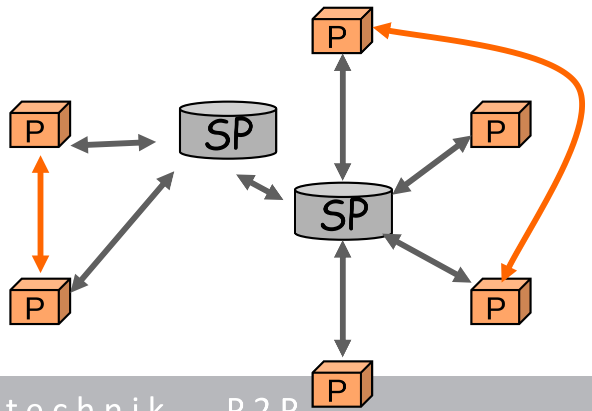
- Napster
- ICQ

- Pure P2P systems
  - Completely de-centralized management and usage of resources



- Gnutella
- DHT applications (Past, I3...)

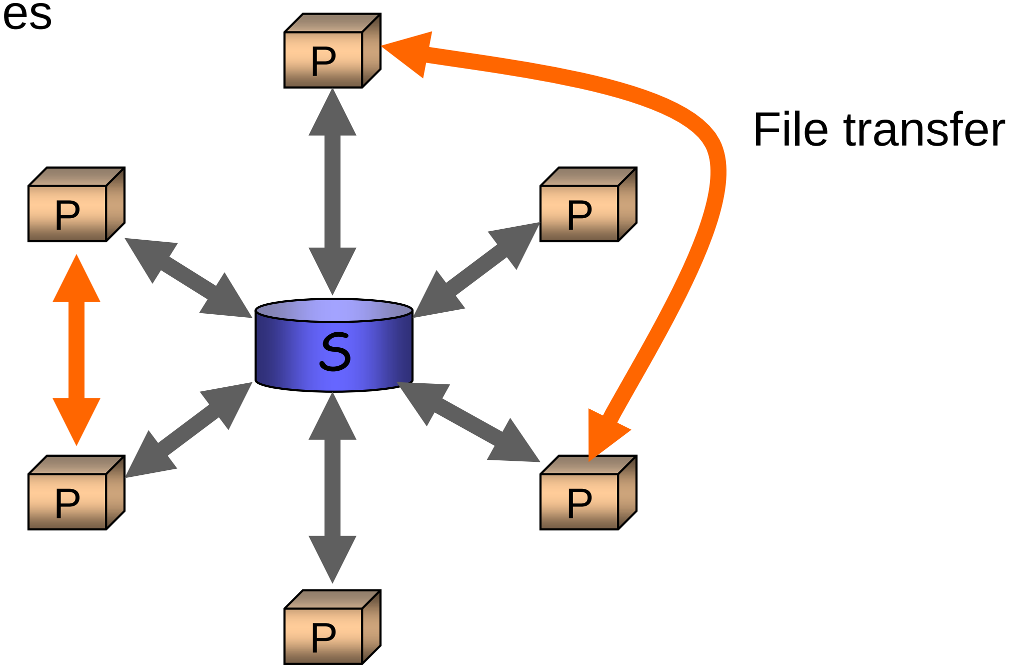
- Hybrid P2P systems
  - Joint usage of distributed resources
  - Direct interaction between peers
  - Some SuperPeers stabilize the Network



- eDonkey
- BitTorrent
- Kazaa

Napster

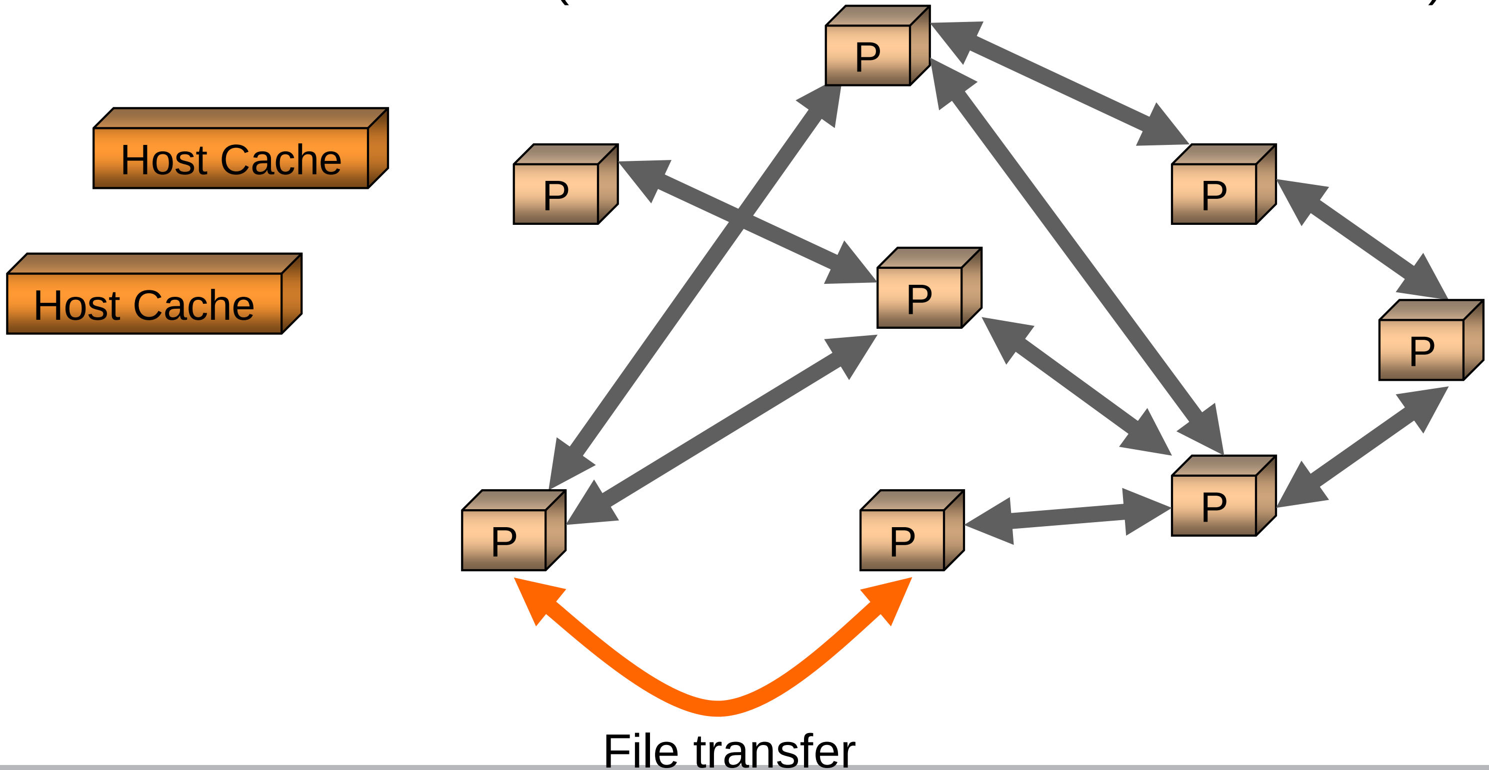
- First P2P killer application (1999-2001): Napster
  - Exchange of MP3 files



  - Central directory server
    - Manages addresses and file lists of peers
    - Does not store files itself

Gnutella

- Decentralized file sharing: Gnutella
  - Host caches provide access points into the P2P network
  - Supports arbitrary file formats
  - Peers are called servents (a combination between server and client)



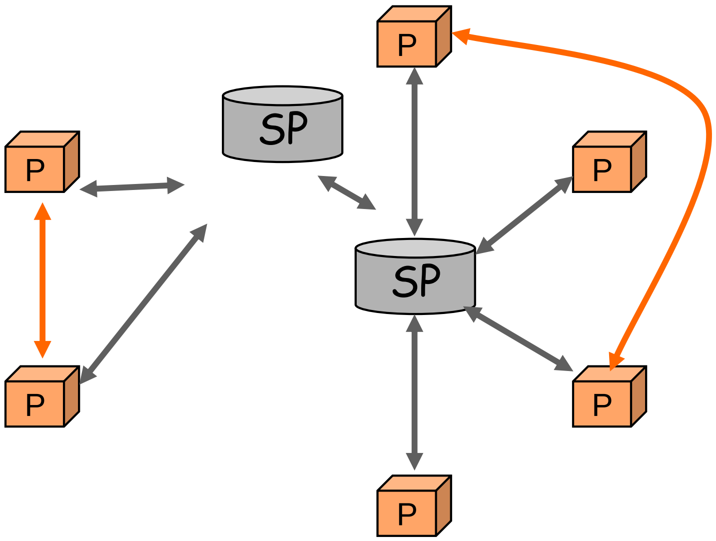
Gnutella

- Peers are connected via TCP connections
- Search requests are flooded over the Gnutella network
  - TCP broadcast of Ping and Query messages
- Discovery of routing loops through pseudo-unique message IDs (UUID)
  - Temporary storage of UUIDs of received messages
  - Discarding of messages that have been received more than once

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

Organization of KaZaA

- Organization
  - Each peer is either itself a super node (SN) or is connected to a SN (bandwidth, better reachability)
  - Each SN knows many other SNs (→ mesh overlay) and behaves like a Napster hub (IP addresses, content) from point of view of their „successors“
  - SNs have on average 200-500 successors
  - > 10.000 SNs possible
  - Supernode list server



BitTorrent - Components: Overview

- BitTorrent
- The BT network is partitioned - many isolated swarms, one swarm per content.
- Many decentralized peers:
  - Leechers: Uploading and Downloading
  - Seeders: Only uploading. Typically has a complete copy of the file
- One centralized node per file: Tracker
  - Coordinates the peers of a swarm

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

BitTorrent - Components: .torrent file

- Meta file used by BT clients to obtain information how to download one file or a batch of files
- Contains:
  - Meta-information about the file (Filename, …)
  - Resource information (tracker, IP of initial seed, …)
- One swarm → one .torrent file
- No build-in search. Search is done on web servers which offer .torrent files, for example: http://thepiratebay.org

BitTorrent - File deployment

- Process:
  - Generate .torrent file with BT client
  - Register .torrent file at public tracker
  - Alternatively run your own tracker
  - Publish .torrent file on web server or similar
  - Seed the file which should be distributed
- If Tracker dies:
  - Peers can not connect to new peers
  - No new peers can join the swarm
  - Peers still can up- and download to each other
  - High probability that the remaining swarm can finish downloading (when its strongly connected, which is most likely the case)
  - The swarm will split up as peers leave
  - The swarm possibly dies

BitTorrent - File downloading: Overview

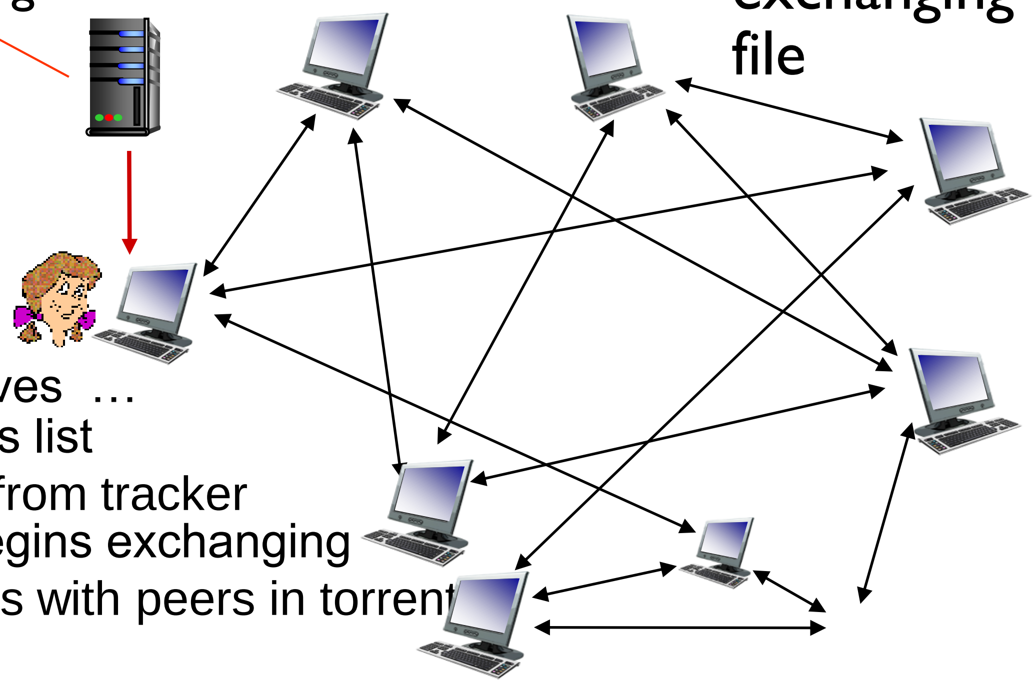
- Process overview:
  - User finds the .torrent file referring to the required download
  - Client registers at a tracker and requests other peers of the swarm
  - Client connects to other peers and down- and uploads the content (leecher)
  - After the download is complete, the leecher becomes a seeder, uploading the content

P2P file distribution: BitTorrent

- File divided into 256Kb chunks
- Peers in torrent send/receive file chunks

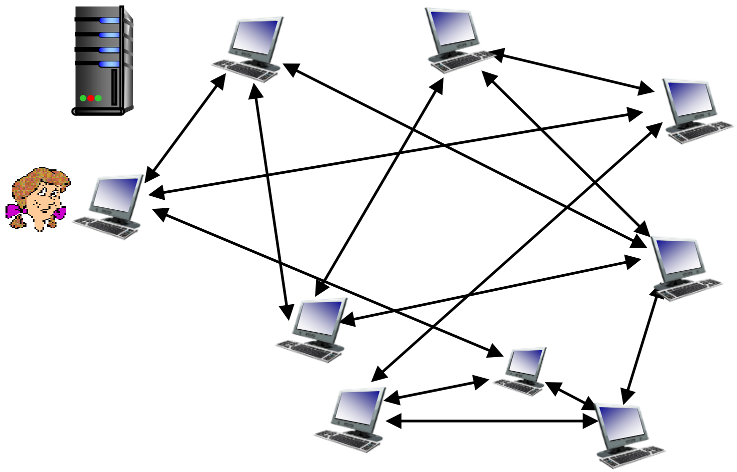
tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



P2P file distribution: BitTorrent

- Peer joining torrent:
  - Has no chunks, but will accumulate them over time from other peers
  - Registers with tracker to get list of peers, connects to subset of peers (“neighbors”)



- While downloading, peer uploads chunks to other peers
- Peer may change peers with whom it exchanges chunks
- Churn: peers may come and go
- Once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent: requesting, sending file chunks

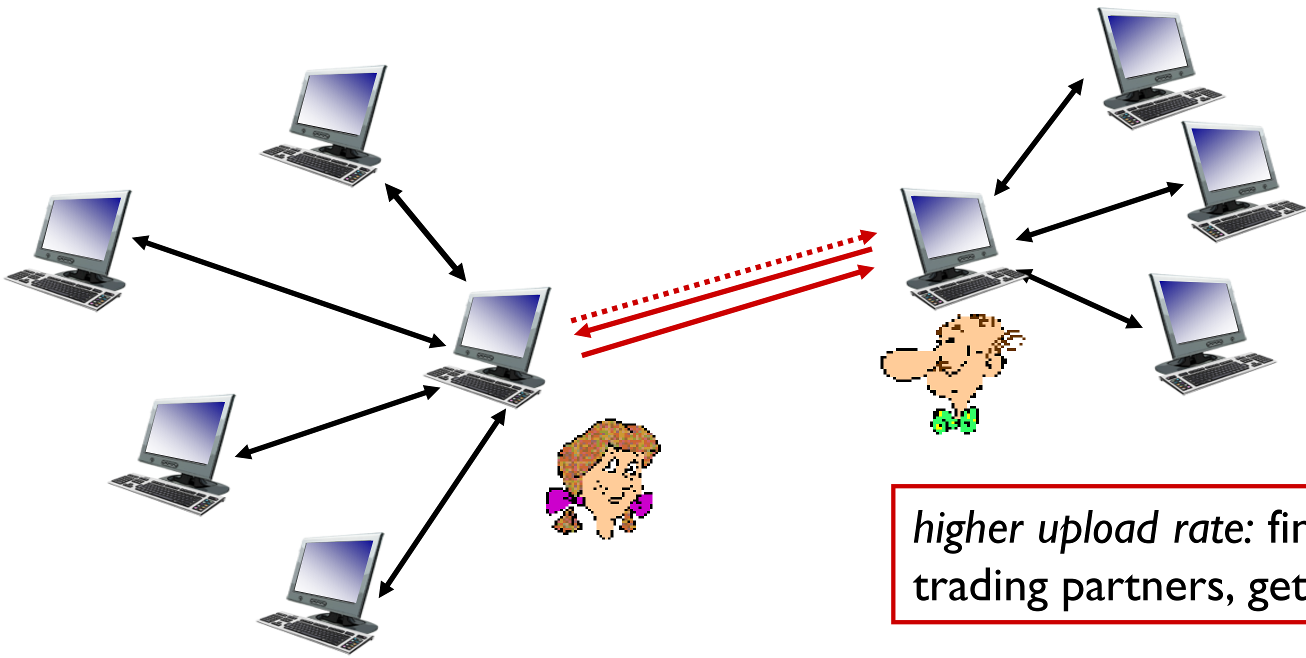
- Requesting chunks:
  - At any given time, different peers have different subsets of file chunks
  - Periodically, Alice asks each peer for list of chunks that they have
  - Alice requests missing chunks from peers, **rarest first**

- Sending chunks: tit-for-tat
  - Alice sends chunks to those four peers currently sending her chunks at highest rate
    - Other peers are choked by Alice (do not receive chunks from her)
    - Re-evaluate top 4 every 10 secs
  - Every 30 secs: randomly select another peer, starts sending chunks
    - “Optimistically unchoke” this peer
    - Newly chosen peer may join top 4

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

BitTorrent: tit-for-tat

- Alice "optimistically unchokes" Bob
- Alice becomes one of Bob's top-four providers; Bob reciprocates
- Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster !

Principle of Distributed Hash Tables

- Principle of distributed hash tables:
  Separation of data & information management

- Nodes have routing informationen (routing tables)
- Routing information provides:
  - Efficient forwarding to „destination“ (content – not location)
  - Statement of existence of content

- Disadvantages:
  - Management of routing information needed
  - Problems with fuzzy requests (e.g, files with name „P*muggle*“)

Principle of Distributed Hash Tables – (cont’d)

- Challenges:
  - (Common P2P requirements) flexibility, reliability and scalability
  - Equal distribution of content to nodes
    - Efficient location of content
  - Permanent adaptation to faults, join, exit of nodes
    - Assignment of responsibilities to new nodes
    - Re-assigment and re-distribution of responsibilities in case of node failure or departure

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

Operation of Distributed Hash Tables

- Form of operation:
- Mapping of nodes and data into a name space (hash domain)
  - Peers and content are addressed using identifiers (IDs)
  - Joint address space
  - Files are assigned to particular peers

- Storage of / search for data at corresponding nodes
  - Search for data = routing to corresponding node
    - Corresponding node not necessarily known in advance
    - Deterministic statement about availability of data

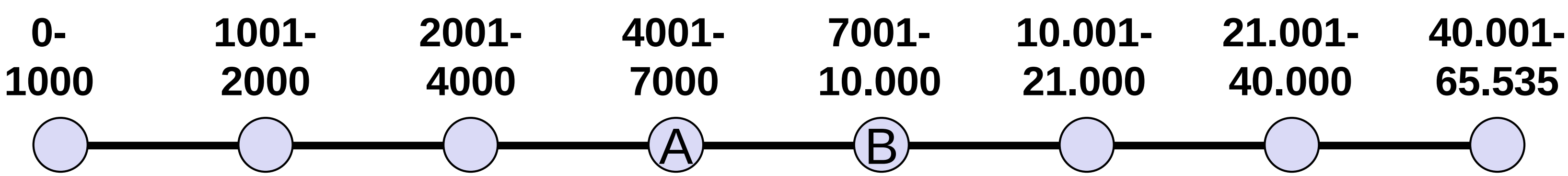
Mapping into Name Space

- Mapping of nodes:
  - Mapping usually by means of hash functions:
    H(“IP address”) or H(“host name”), respectively
    e.g., H(www.cnn.com) = 4001, H(134.2.11.140) = 4712
- Mapping of data:
  - Mapping also usually by means of hash functions:
    H(“file name”) or H(“content”), e.g., H(“movie”) = 2313
  - Storage in form of pairs (key, value)
    (2313, movie) or (2313, storage location of movie)
  - ID of data can be independent of content (different semantics)

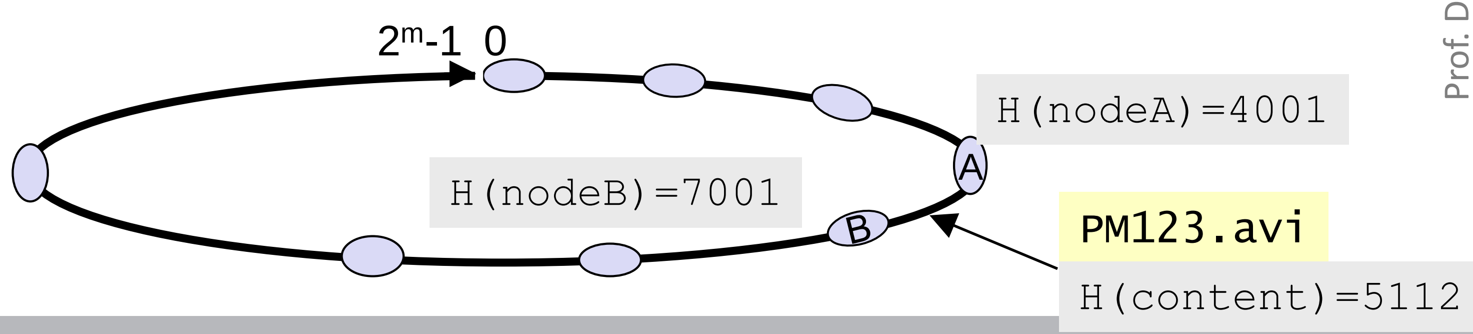
Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

4.1 Operation – Step 1: Mapping into Range

- Step 1:
  Mapping of content/nodes into linear range
  - Usually: 0, …, 2m-1 >> number of objects to be stored
  - Mapping of content and nodes into value range by means of a hash function
    - E.g. H(„/movie/Matrix/divx/en“) → 2313
- Distribution of value range over DHT nodes („responsibilities“)



Value range often represented as a ring

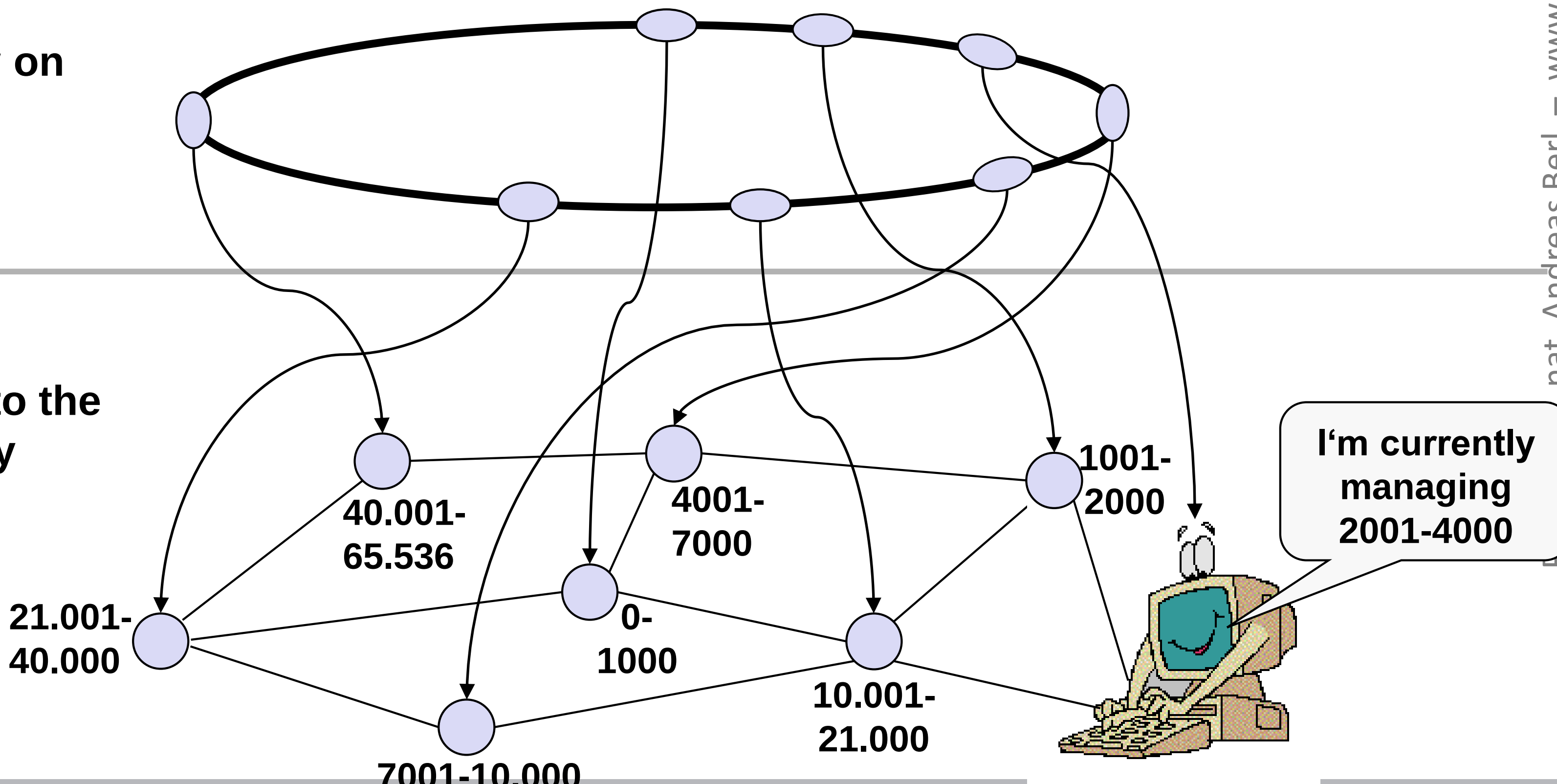


4.1 Distribution of Value Range Across DH Nodes

- Each node is responsible for part of the value range
  - Often highly redundant
  - Repeated adaptation

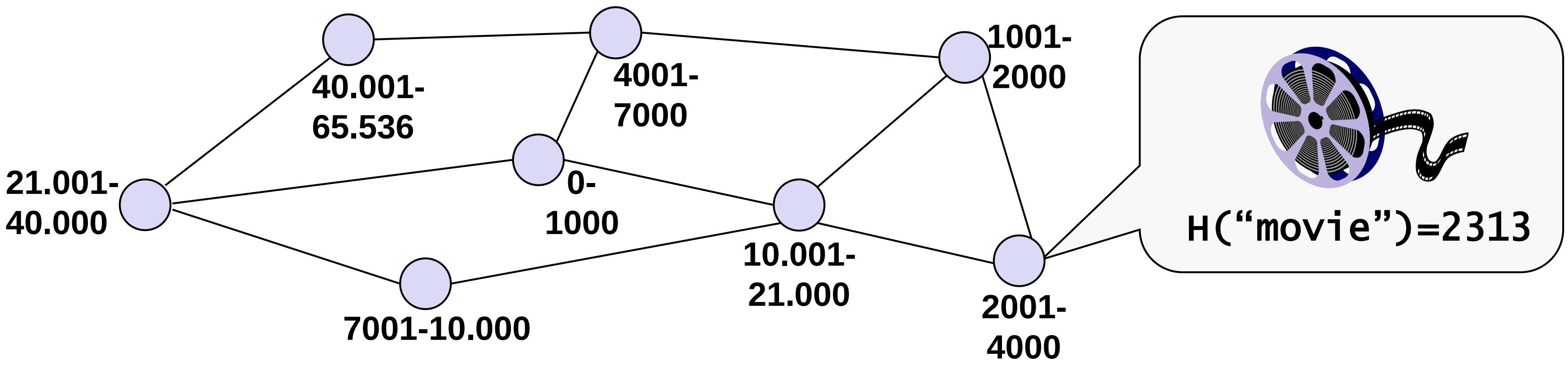
Logical view on distributed hash tables

Mapping onto the real topology



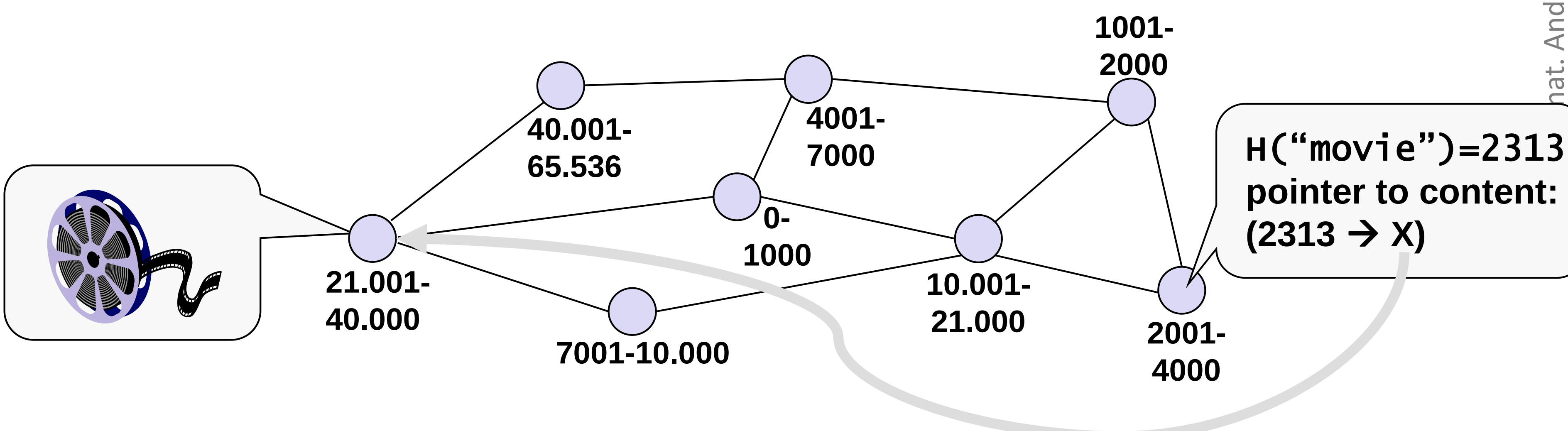
4.1 Storage of Content with DHTs

- How is content stored on the nodes?
  - Assumption: H(“movie”) = 2313 is mapping into value range
- Direct storage:
  - Content is stored on node H(“movie”), i.e, copied to 2313
    → inflexible for large content – o.k. if not too much data



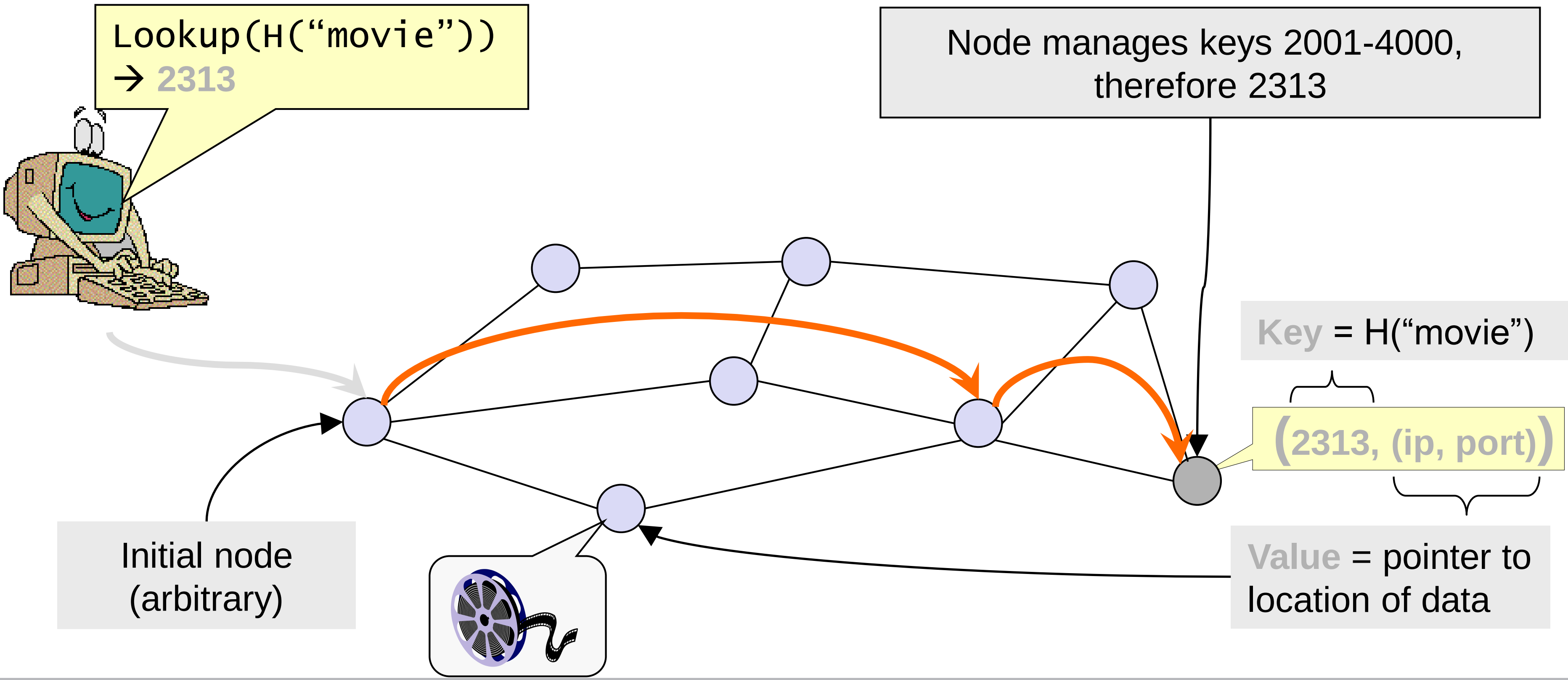
4.1 Storage of Content with DHTs (cont`d)

- Indirect storage:
  - Nodes in DHT store pair (key,value)
    - Key = Hash(„movie”)→ 2313
    - Value is often real storage address of content: (IP, Port) = (134.2.11.140, 4711)
  - More flexible, but one step more to reach content



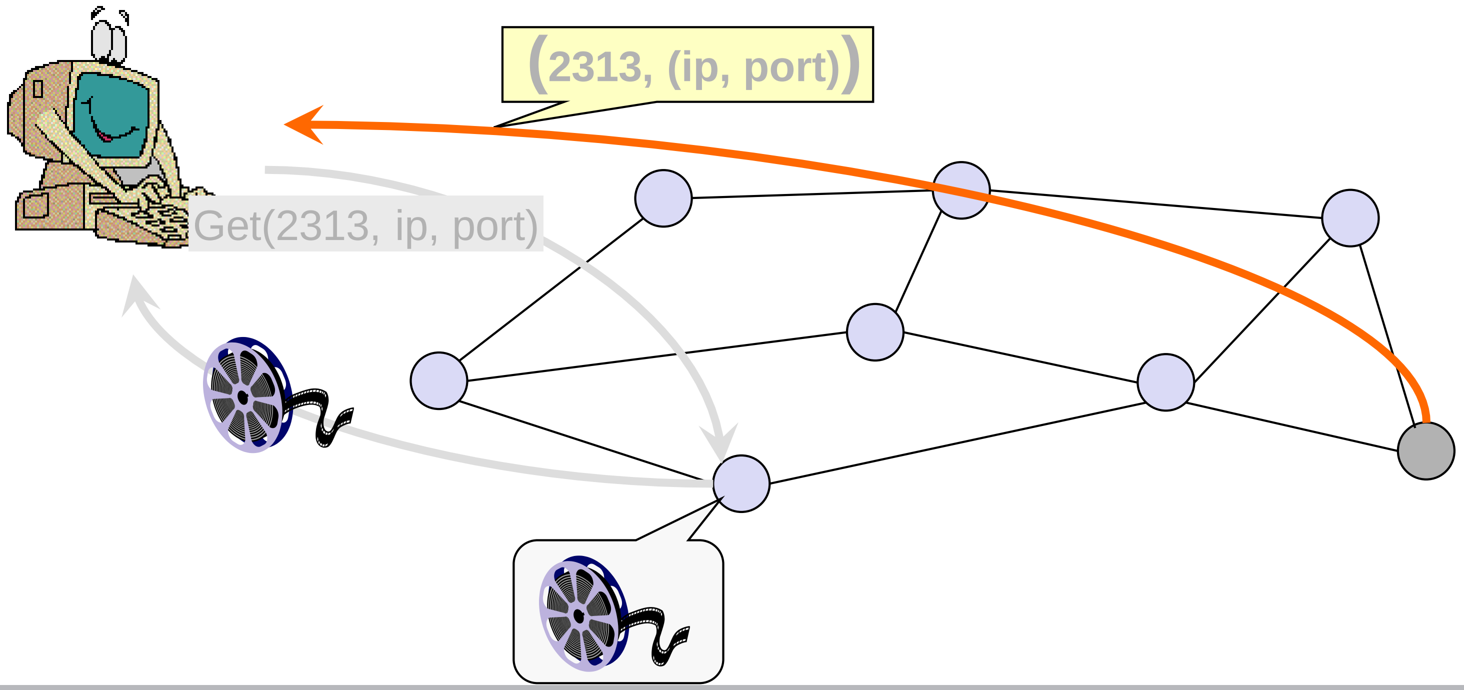
4.1 Operation – Step 2: Routing to Data (cont`d)

- Search for key
  - Start at arbitrary initial node
  - Routing to requested content (Key)



4.1 Location of Content

- Delivery of information pair (key, value)
- Evaluation and request for real content



4.1 DHT Updating: Joining and Exit of Nodes

- Joining of a new node
  - Assignment of a particular hash range
  - Copying of K/V pairs of hash range (usually with redundancy)
  - Binding into routing environment
- Exit of a node
  - Partitioning of hash range to neighbor nodes
  - Copying of K/V pairs to corresponding nodes
  - Unbinding from routing environment
- Failure of a node
  - Use of redundant K/V pairs (if a node fails)
  - Use of redundant / alternative routing paths
  - Key-value usually still reachable if at least one copy remains

4.1 Resume: Distributed Hash Tables

- Resume: properties of DHTs
  - Use of routing information for efficient search for content
  - Keys are evenly distributed across nodes of DHT (usually with redundancy incorporated)
    - No bottlenecks
    - A continous increase in number of stored keys is admissible
    - Failure of nodes can be tolerated
    - Survival of attacks possible
  - Self-organizing system
  - Simple and efficient realization
  - Supporting a wide spectrum of applications
    - Flat (hash) key without semantic meaning
    - Value depends on application

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

DHT Approaches

- Examples of distributed hash tables
  - Chord
    UC Berkeley, MIT
  - Pastry
    Microsoft Research, Rice University
  - Tapestry
    UC Berkeley
  - CAN
    UC Berkeley, ICSI
  - P-Grid
    EPFL Lausanne
  - Nice, ZigZag
- And many more: Kademlia, Symphony, Viceroy, …

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

Chord

- Simple database with(key, value) pairs:
  - Key: movie title; value: IP address
  - Key: human name; value: social security #

Key	Value
John Washington	132-54-3570
Diana Louise Jones	761-55-3791
Xiaoming Liu	385-41-0902
Rakesh Gopal	441-89-1956
Linda Cohen	217-66-5609
…….	………
Lisa Kobayashi	177-23-0199

Chord: Hash Table

- More convenient to store and search on numerical representation of key
- Key = hash(original key)

Original Key	Key	Value
John Washington	8962458	132-54-3570
Diana Louise Jones	7800356	761-55-3791
Xiaoming Liu	1567109	385-41-0902
Rakesh Gopal	2360012	441-89-1956
Linda Cohen	5430938	217-66-5609
…….		………
Lisa Kobayashi	9290124	177-23-0199

Chord

- Distribute (key, value) pairs over millions of peers
  - Pairs are evenly distributed over peers
- Any peer can query database with a key
  - Database returns value for the key
  - To resolve query, small number of messages exchanged among peers
- Each peer only knows about a small number of other peers
- Robust to peers coming and going (churn)

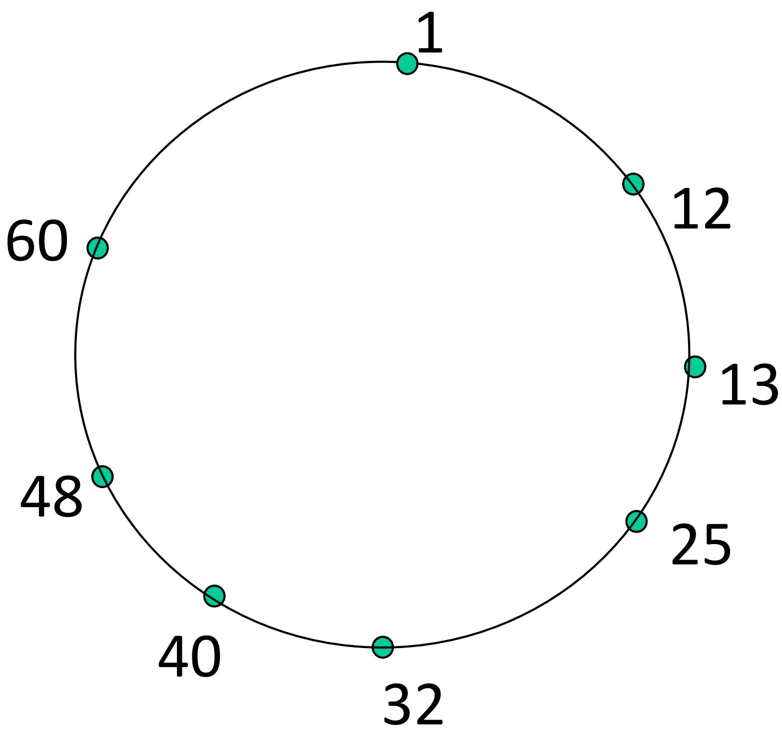
Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

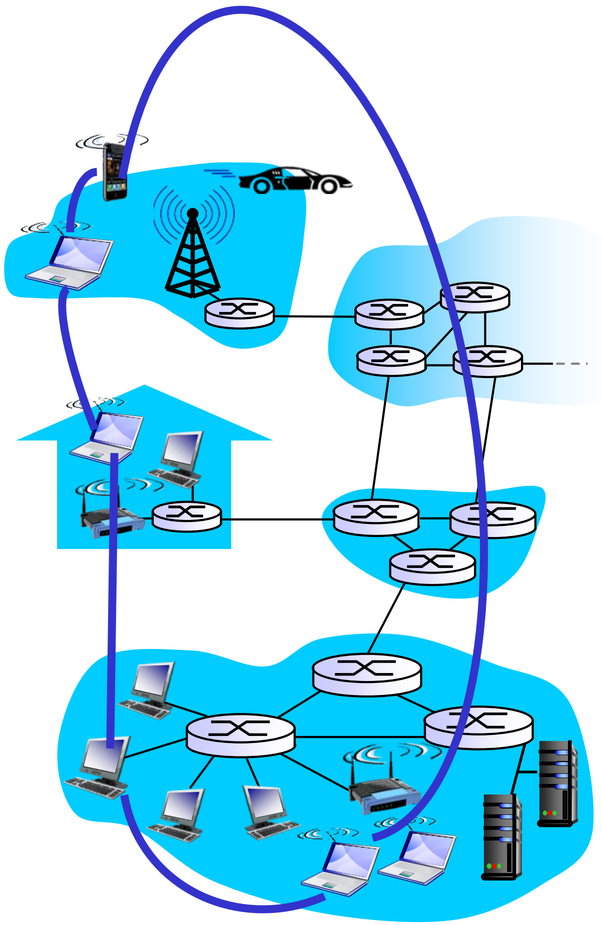
Chord: Assign key-value pairs to peers

- Rule: assign key-value pair to the peer that has the closest ID.
- Convention: closest is the immediate successor of the key.
- E.g., ID space {0,1,2,3,…,63}
- Suppose 8 peers: 1,12,13,25,32,40,48,60
  - If key = 51, then assigned to peer 60
  - If key = 60, then assigned to peer 60
  - If key = 61, then assigned to peer 1

Chord

- Each peer only aware of immediate successor and predecessor.

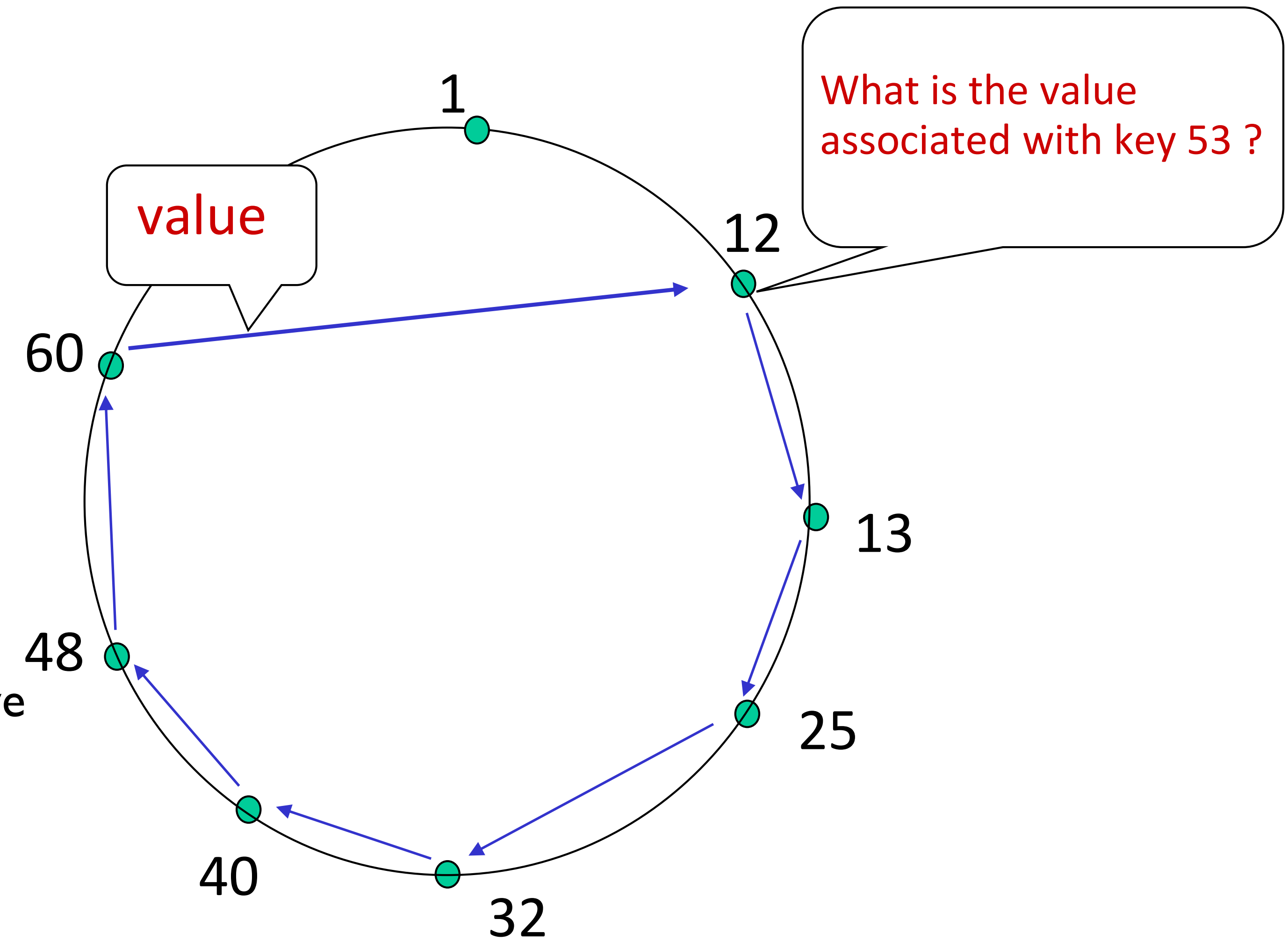




“overlay network”

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

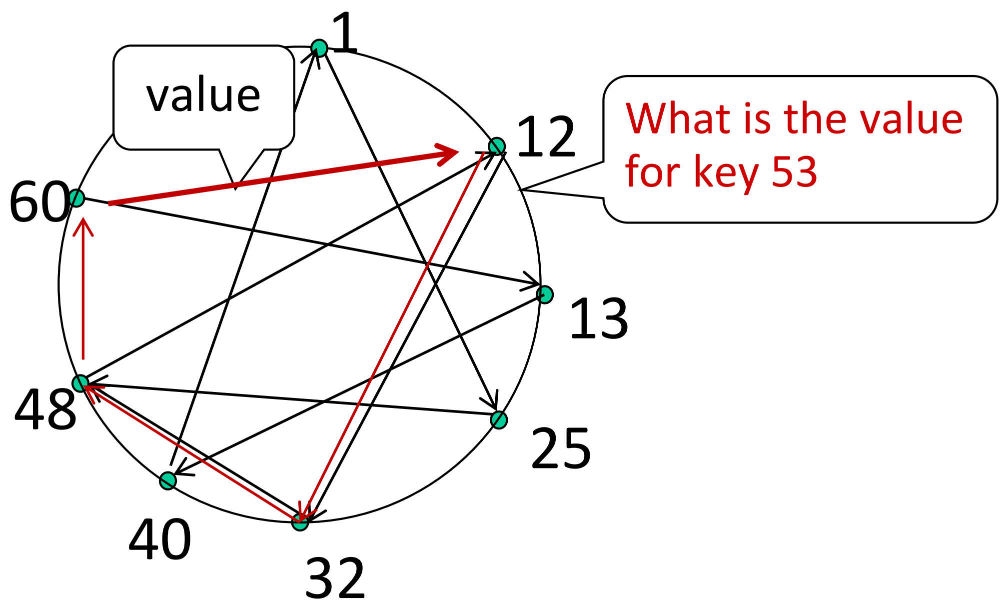
Chord: Resolving a query



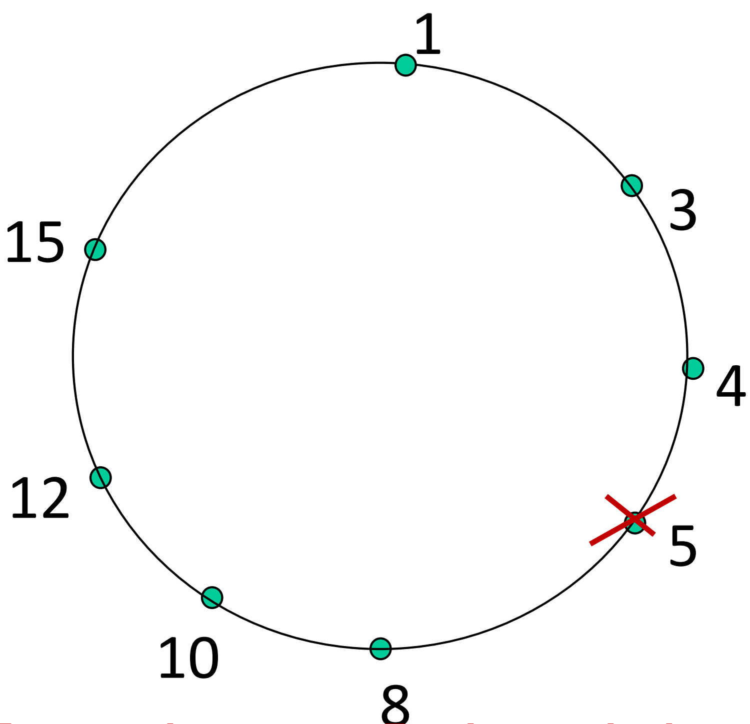
O(*N*) messages on average to resolve query, when there are *N* peers

Chord: shortcuts

- Shortcuts
  - Possible to design shortcuts with O(log N) neighbors, O(log N) messages in query



Chord: Peer churn

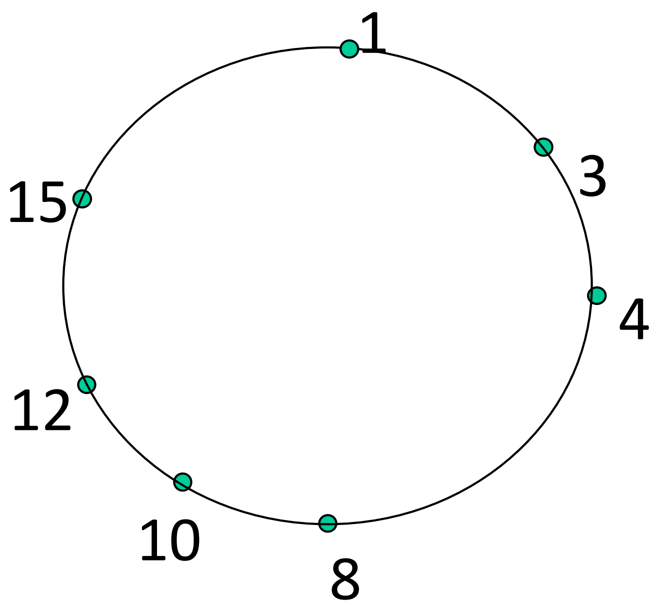


Example: peer 5 abruptly leaves

- Handling peer churn:
  - Peers may come and go (churn)
  - Each peer knows address of its two successors
  - Each peer periodically pings its two successors to check aliveness
  - If immediate successor leaves, choose next successor as new immediate successor

Prof. Dr. rer. nat. Andreas Berl – www.andreas-berl.de

Chord: Peer churn



- Handling peer churn:
  - Peers may come and go (churn)
  - Each peer knows address of its two successors
  - Each peer periodically pings its two successors to check aliveness
  - If immediate successor leaves, choose next successor as new immediate successor

- Peer 4 detects peer 5's departure; makes 8 its immediate successor
- 4 asks 8 who its immediate successor is; makes 8's immediate successor its second successor.

Chord

- Complexity
  - Search overhead: O(log N)
  - Storage requirement per node: O(log N)
  - Fault recovery cost: $O(\log^2 N)$
- Advantages
  - Complexity can be theoretically derived
  - Logarithmic overhead maintained in the face of node failures
- Disadvantages
  - No explicit proximity search for near-by nodes
  - Security has not yet been taken fully into account

Summary

- Centralized, Decentralized and Hybrid P2P
  - Search overhead: O(log N)
  - Storage requirement per node: O(log N)
  - Fault recovery cost: $O(\log^2 N)$
- Advantages
  - Complexity can be theoretically derived
  - Logarithmic overhead maintained in the face of node failures
- Disadvantages
  - No explicit proximity search for near-by nodes
  - Security has not yet been taken fully into account