b

UNIVERSITÄT
BERN

b
UNIVERSITÄT
BERN

Advanced Networking and Future Internet

III. Peer-to-Peer Networks

Prof. Dr. Torsten Braun, Institut für Informatik

Bern, 28.09.2020

b UNIVERSITÄT BERN

Advanced Networking and Future Internet: Peer-to-Peer Networks

Table of Contents

1. Introduction
    1. Client/Server vs. Peer-to-Peer Networks
    2. Peer-to-Peer Applications
    3. DNS vs. Peer-to-Peer Networks
    4. Centralized and Distributed Peer-to-Peer-Networks
    5. Distributed Peer-to-Peer Networks
2. Unstructured Peer-to-Peer Networks
    1. Distribution of Query Messages
    2. Gnutella
    3. Hierarchical P2P Networks: Kazaa
    4. BitTorrent
3. Structured Peer-to-Peer Networks: Distributed Hash Tables
    1. Single and Distributed Hash Tables
    2. Consistent Hashing
    3. Chord
    4. Content Addressable Network
    5. Tapestry (Plaxton) Routing
    6. Pastry
    7. Kademlia

1. Introduction

1. Client/Server versus Peer-to-Peer Networks

Client/Server Paradigm

- Powerful server nodes and rather simple client nodes
- Example: World Wide Web
- Central data storage
- High availability of servers

Peer-to-Peer (P2P)

- Equal systems
- Client and server functionality
- Decentralized, distributed data storage
- Sharing of computing and storage resources
- Symmetrical end-to-end communication
- Autonomous, self-controlling nodes
- Nodes do not need to be on-line permanently. No permanent IP addresses required
- Peers form and self-organize overlay network.
- Often high churn rate of peers

1. Introduction

2. Peer-to-Peer Applications

- Distributed lookup service
- Distributed file systems
- Distributed network measurement
- Overlay network for application level multicast distribution
- TV and audio/video streaming
- Content delivery networks to avoid flash crowds
- Grid computing
- Collaboration tools and groupware
- Communications: conferencing, instant messaging
- Cryptocurrencies

1. Introduction

3. DNS vs. Peer-to-Peer Networks

Conventional DNS mapping

– logical name (e.g., host name) $\rightarrow$ IP address

– Mapping is *centralized.*

Peer-to-Peer Networks

– Application specific mapping: application ID (e.g., file ID) $\rightarrow$ IP address

– Mapping may be *distributed.*

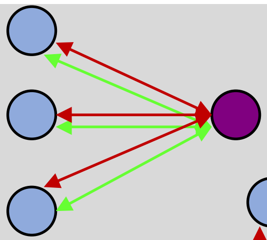
1. Introduction

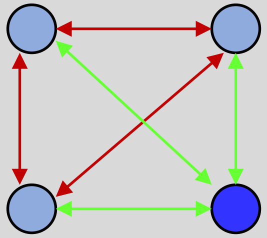
4. Centralized and Distributed Peer-to-Peer-Networks

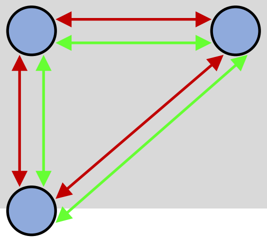
- Index and file server
  - contains file index and corresponding files
- P2P network with central index server
  - contains file index
  - Files are distributed among peers.
  - Example: Napster
- Distributed P2P network
  - Index and files are distributed among peers.

  → Redundancy

  → Load balancing
  - Option: some nodes (aka super nodes) have special roles.







1. Introduction

5. Distributed Peer-to-Peer Networks

Unstructured

- Characteristics
  - Flat network without structure
  - Placement of data unrelated to overlay topology
  - Random searches
- Advantages
  - Easy accommodation of transient nodes
  - High fault tolerance
  - Keyword search possible
- Disadvantages
  - Inefficient / not scalable searches, e.g., flooding, random walk
  - Possible improvement by replication
- Examples
  - Gnutella
  - KaZaa / FastTrack
  - BitTorrent

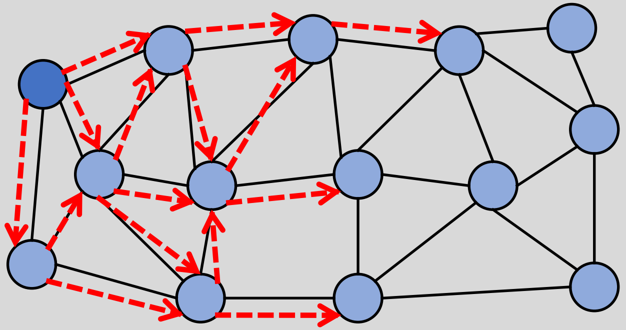
Structured

- Characteristics
  - Establishment of overlay network structure (TCP connections or IP address pointers)
  - Mapping of keys to nodes
- Advantages
  - Rather efficient searches
  - Scalability
- Disadvantages
  - Limited fault tolerance
  - Overhead for joins / leaves
  - Difficult keyword search
- Examples
  - Chord
  - Content Addressable Network
  - Tapestry
  - Pastry

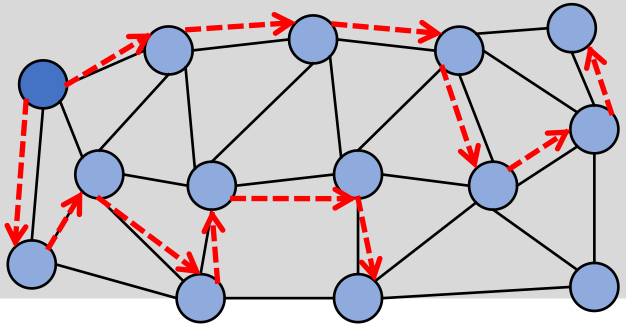
2. Unstructured Peer-to-Peer Networks

1. Distribution of Query Messages

– Flooding / expanding ring search
  – Example: TTL = 3



– Random walk
  – Multiple parallel random walks
  – Example: TTL = 6

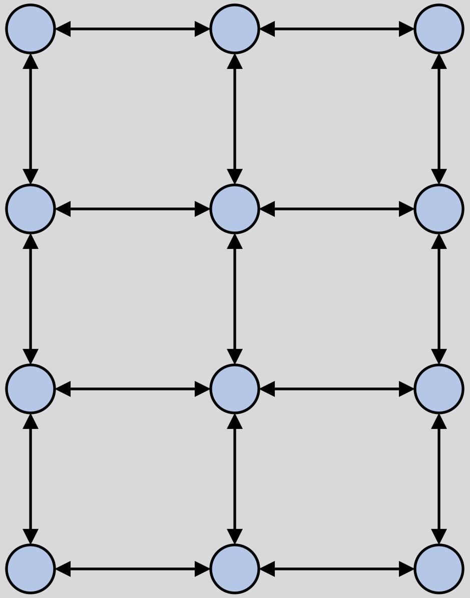


UNIVERSITÄT BERN

2. Unstructured Peer-to-Peer Networks

2. Gnutella

- P2P node = “servent” (server + client)
- Establishment of initial connections
  - Connection to a priori known nodes
  - Host caches in permanently running nodes return lists of other nodes to be contacted.
  - Each host maintains own list of hosts.
- Communication mechanisms
  - Application level routing of messages
  - TCP broadcasts to neighbors
- Messages include
  - Globally unique identifier
    - Loop control
    - Back-propagation of responses
  - Time to live value (TTL)



2. Unstructured Peer-to-Peer Networks

2.1 Gnutella Messages

- PING / PONG
  - Newly entering node broadcasts PING to announce its existence.
  - Response from neighbor including IP address and other information
- QUERY / QUERY HIT
  - QUERY messages for file searching includes search string for file.
  - Receiving node compares search string with available files.
  - QUERY HIT includes IP address of peer.
- PUSH REQUEST
  - Normal case: file download using HTTP
  - PUSH REQUEST (if server is behind a firewall) asks server to establish a connection to the client.

2. Unstructured Peer-to-Peer Networks

2.2 Gnutella Problems

- Flooding of control messages (PING, QUERY)
  - 95 % of all node pairs have a hop distance of less than 7 hops (default TTL value = 7 !).
  - can easily use full bandwidth of a low bandwidth peer.
  - Network collapsed in 2000 (network fragmentation), because peers with bad connectivity could not handle control message load.
- Mismatch of Gnutella and IP network
  - 2-5 % of Gnutella connections interconnect nodes within the same autonomous system (AS), although > 40 % of nodes are within top 10 ASs.
  - unnecessarily high number of traffic crossing ASs
- Free Riding
  - Many nodes are connected for a few hours only.
  - 50 % of requests are answered by 1 % of nodes.

2. Unstructured Peer-to-Peer Networks

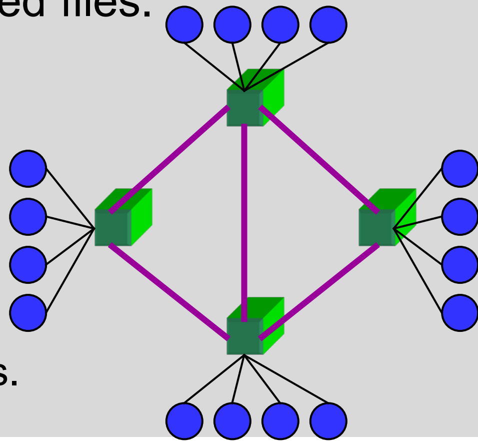
2.3 Gnutella Optimizations

- Reflector nodes
  - Caching of QUERY HIT messages
  - Subsequent QUERY messages can be answered by reflector.
- Proactive replication of objects and random forwarding of QUERY messages
- Distributed flow control
  - Nodes store and exchange information about current and tolerable load (incoming / outgoing messages)
  - Redirection of requests to other less loaded nodes in case of overload
- Establishment of efficient overlay network topologies by considering underlying topology
- Hierarchical network organization, super nodes, cf. Kazaa
- Bootstrapping: pre-configured peers, host caches on web pages

2. Unstructured Peer-to-Peer Networks

3. Hierarchical P2P Networks: Kazaa

- Introduction of multiple, e.g., 2, levels: super peers and ordinary peers, cf. Kazaa
- Every peer is either a super peer or an ordinary peer assigned to a super peer.
- Bootstrap: Ordinary peers connect to super peers (obtained from bootstrap server) and upload meta data for shared files.
- Super peers
  - nodes with fast Internet connections and high processing power
  - organize themselves on the upper level and protect ordinary peers from traffic.
  - form a Napster-like hub for ordinary peers
  - know where other super peers are located (partial mesh structure).
  - might forward queries from own ordinary peers to other super peers.
  - answer queries from own ordinary peers and other super peers.



2. Unstructured Peer-to-Peer Networks

4.1 BitTorrent

– Files are divided into pieces, which are distributed over several peers.

→ avoiding bottlenecks with one client storing the whole file

– File is stored by a node set called swarm.

– Joining to swarms
  – Client downloads meta-data about files and swarms from a web server, e.g., file name, file size and tracker of swarm (.torrent file).
  – Client joins the swarm by sending a request message to tracker.
  – Tracker responds to client request with a partial list of peers in the swarm.

– Nodes hosting files create and store .torrent file and start trackers.

UNIVERSITÄT BERN

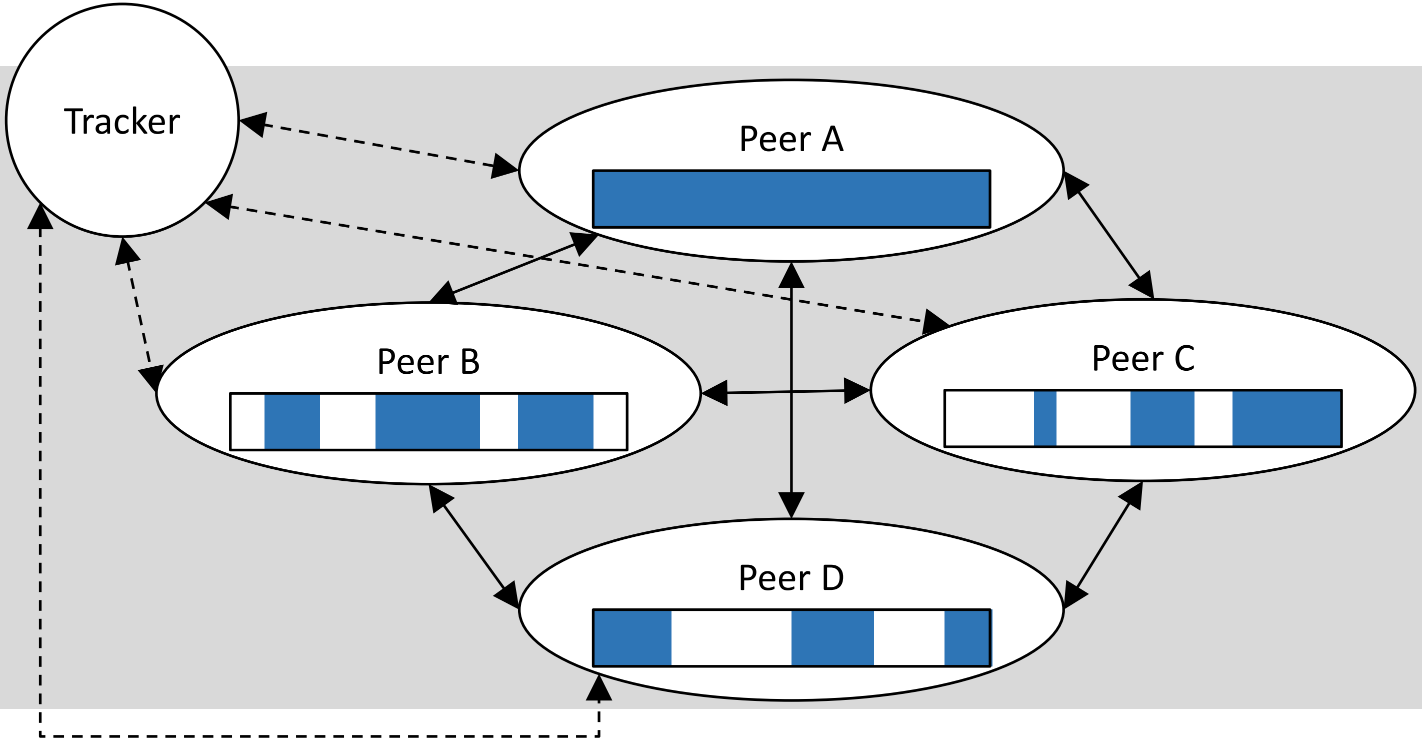
2. Unstructured Peer-to-Peer Networks

4.2 BitTorrent

- File download: Client
  - establishes TCP connections to some peers of the swarm
  - exchanges / updates information about available pieces
  - requests pieces randomly and in random order (rarest first)
  - updates piece information to peers after download
- Tit for tat to achieve fairness
  - Temporary choking (stop of uploads) of bad (only downloading) peers, periodic unchoking
  - Selection of peers for upload based on download rate
- Problem:
  Tracker as single point of failure
  → distribution of .torrent file by structured peer-to-peer networks
  → “trackerless” versions: overlay network of peer finders

2. Unstructured Peer-to-Peer Networks

4.3 BitTorrent File Download

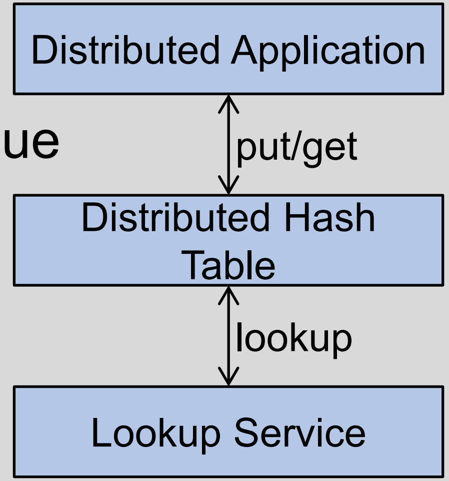


3. Structured Peer-to-Peer Networks: Distributed Hash Tables

1. Single and Distributed Hash Tables

Single node hash table

- key = hash (data)
- data structure with mappings: key → value
- Interface:
  - put (key, value)
  - get (key) → value



Distributed hash table

- key = hash (data)
- lookup (key) → node_address
- Routing of put and get messages
  - route(node_address, put, key, value)
  - route(node_address, get, key) → value

3. Structured Peer-to-Peer Networks: DHTs

2. Consistent Hashing

Mapping of both objects and node (IP) addresses to the same ID space; objects are stored at closest nodes in ID space.

– hash(object_name) → object_id

– hash(node_address) → node_id

Examples:

– Chord

– Content Addressable Network

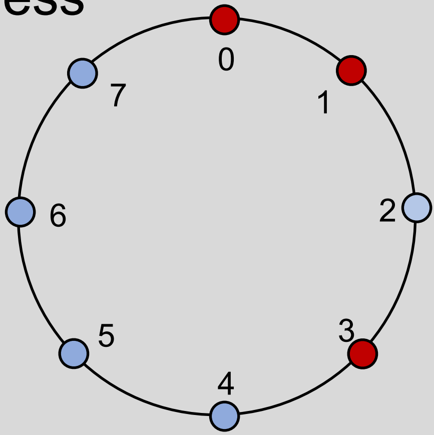
– Tapestry

– Pastry

3. Structured Peer-to-Peer Networks: DHTs

3. Chord

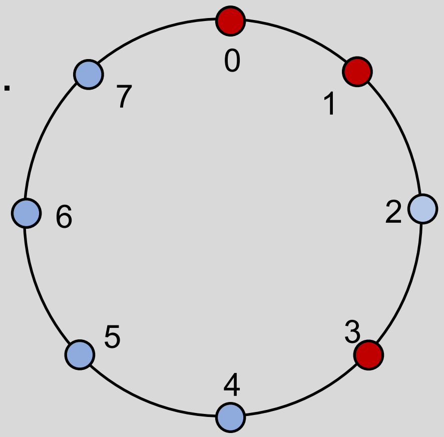
- Chord provides a lookup service: lookup(key) → IP address
- Keys and nodes are mapped to m-bit identifiers by consistent hashing.
- Identifiers are ordered in a circle modulo 2^m, e.g. $m = 3$
- Keys are assigned to the node with the same identifier or to the first following node.
  - Example: 1 → 1, 2 → 3, 6 → 0
- Joining and leaving nodes requires to move 1/N keys between nodes.



3. Structured Peer-to-Peer Networks: DHTs

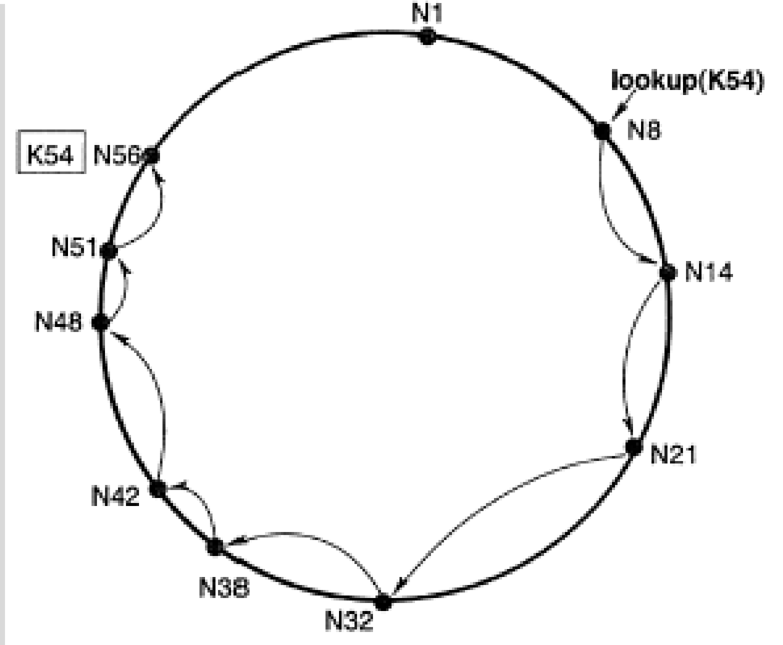
3.1 Chord: Routing

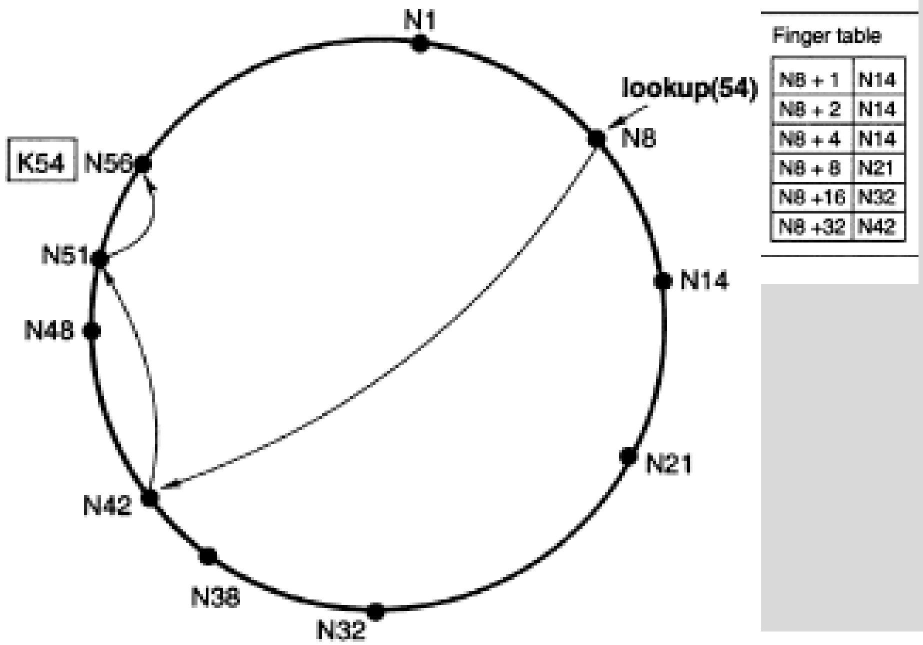
- Naive: from one node to the next
- Support by special routing tables (finger tables) at node n
  - i^{th} entry: node s that succeeds n by at least 2^{i-1}
  - Node n knows more about close nodes than about far nodes.
  - Finger tables:
    - at node 0: $0 + 1 = 1 \rightarrow 1$; $0 + 2 = 2 \rightarrow 3$; $0 + 4 = 4 \rightarrow 0$
    - at node 1: $1 + 1 = 2 \rightarrow 3$; $1 + 2 = 3 \rightarrow 3$; $1 + 4 = 5 \rightarrow 0$
    - at node 3: $3 + 1 = 4 \rightarrow 0$; $3 + 2 = 5 \rightarrow 0$; $3 + 4 = 7 \rightarrow 0$
  - A node forwards messages to the node closest to the target.



3. Structured Peer-to-Peer Networks: DHTs

3.2 Chord: Naive and Finger Table Routing





Finger table

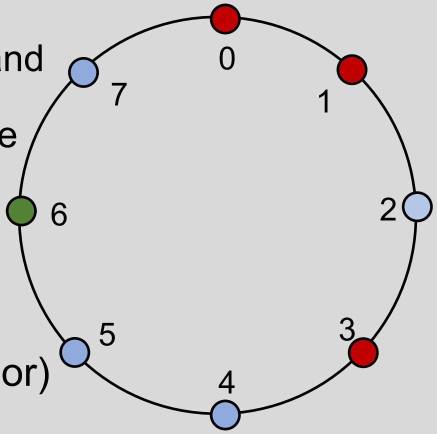
N8 + 1	N14
N8 + 2	N14
N8 + 4	N14
N8 + 8	N21
N8 +16	N32
N8 +32	N42

UNIVERSITÄT BERN

3. Structured Peer-to-Peer Networks: DHTs

3.3 Chord: Node Joins and Leaves

- Nodes maintain predecessor pointer.
- Node n joins the network
  1. Initialisation of predecessor and finger table of node n
     - Node n learns some node m and asks m to look up the predecessor and finger tables of n.
     - Option: contact a direct neighbour, copy its (similar) table, correct table
  2. Update of predecessor and finger table at existing nodes
  3. Notification of higher layer software to transfer keys from the joining node's successor
- Successor list (r nearest successors of a node)
  - for redundancy (failed successor can be replaced by first live successor)
  - can be used for data replication.



3. Structured Peer-to-Peer Networks: DHTs

4. Content Addressable Network

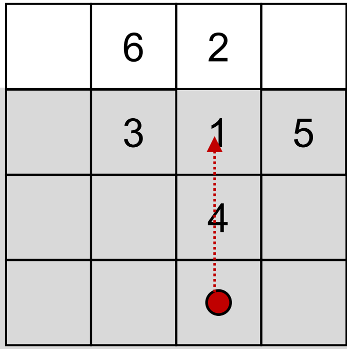
- For storing a key-value pair (k_1,v_1): k_1 is mapped to a point p in a d-dimensional coordinate space.
- (k_1,v_1) will be stored by the node that owns the zone, in which p lies.
- Each node maintains a routing table with entries for its immediate neighbours in the coordinate space (IP address, virtual zone).
- 2d routing table entries at each node (d: dimensions)
- Greedy forwarding to neighbour closest to the destination (RTT can be considered, if multiple neighbours exist.)
- Average routing path length: $(d/4)\ (n^{1/d})$

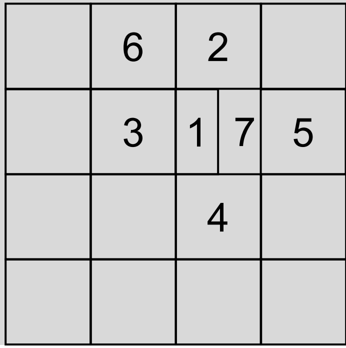
UNIVERSITÄT BERN

3. Structured Peer-to-Peer Networks: DHTs

4.1 CAN Construction

– Join of a new node
  1. New node must find an existing CAN node.
  2. New node must find (using CAN routing mechanisms) a node whose zone will be split: selection of random point p, JoinRequest for p, zone split
  3. Neighbours of split zone must be notified to include new node into routing.
  4. Handover of key-value pairs to new node
– Takeover mechanism for departing nodes
– Replication
  – Construction of multiple coordinate spaces (realities)
  – Replication of data in each reality
– Zone overloading
  – A zone is shared by MAXPEERS peers.
  – Distribution or replication of key-value pairs among peers





3. Structured Peer-to-Peer Networks: DHTs

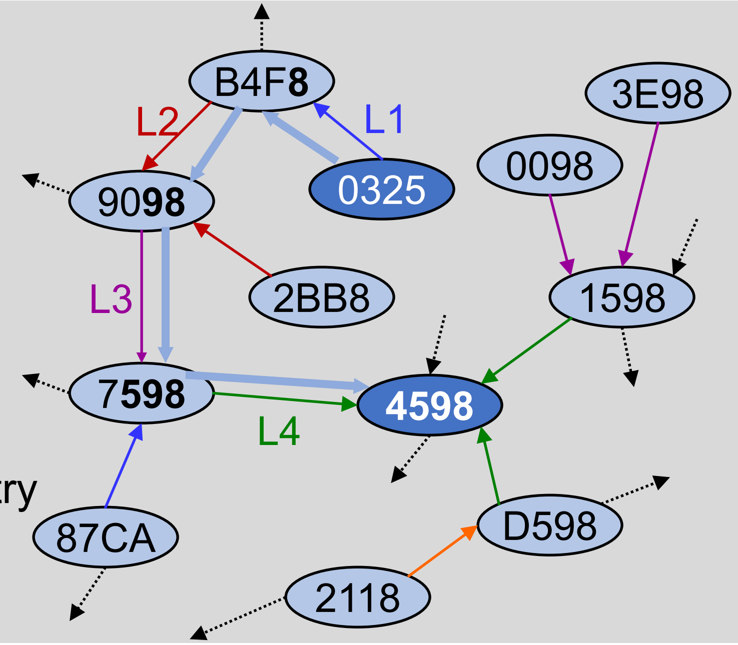
4.2 CAN Optimization

- Neighbours of a peer can be very far away.
- Attempt to construct CAN topology congruent with IP topology
- Landmarks:
  well-known set of machines
- Node measures RTT to landmarks and orders landmarks in order of increasing RTTs.
- m landmarks → m! (= m · (m-1) · (m-2) · … · 1) possible orderings
- Partitioning of space into m! equally sized portions
- A node joins at a random point within the portion associated with the landmark ordering.
- Close nodes will join the same portion.
- Problem: non-uniform population
- Solution: load balancing by hand-off of portion parts

3. Structured Peer-to-Peer Networks: DHTs

5.1 Tapestry (Plaxton) Routing

- Routing table with multiple levels, where each level has entries with equal matching suffixes.
- i^{th} entry of j^{th} level at node N: ID and location of the closest node that ends in „i“ + suffix (N, j-1)
- Example: node 9098
  - 3rd level, 5th entry: node closest to 9098 ending in 598: 7598
  - 4 levels, 16 entries per level $\rightarrow$ 64 entries in routing table
- Name space N, base b: Destination can be reached with $\log_b N$ hops.
- 3 closest candidates (in network latency) for each entry
- Surrogate routing in case of empty entries (next highest entry)



3. Structured Peer-to-Peer Networks: DHTs

5.2 Tapestry: Dynamic Node Algorithms

- Joining Nodes
  - select random NodeIDs
  - connect to Tapestry node close to itself
  - learn routing tables from other nodes
  - connect to newly learned nodes as neighbors
- Voluntary Node Deletion
  - Notification of neighbors about leave and replacement nodes in routing table
- Involuntary Node Deletion
  - Backup forwarding entries in routing table

3. Structured Peer-to-Peer Networks: DHTs

6. Pastry

– Random assignment of node IDs between 0 and 2^{128}-1

– Messages are routed to node with ID closest to destination.

Each node has

– Routing table
  – 2^b-1 entries in $\log_{2^b} N$ rows (N: number of nodes), e.g., b = 4
  – Entries in row n share first n digits with node.

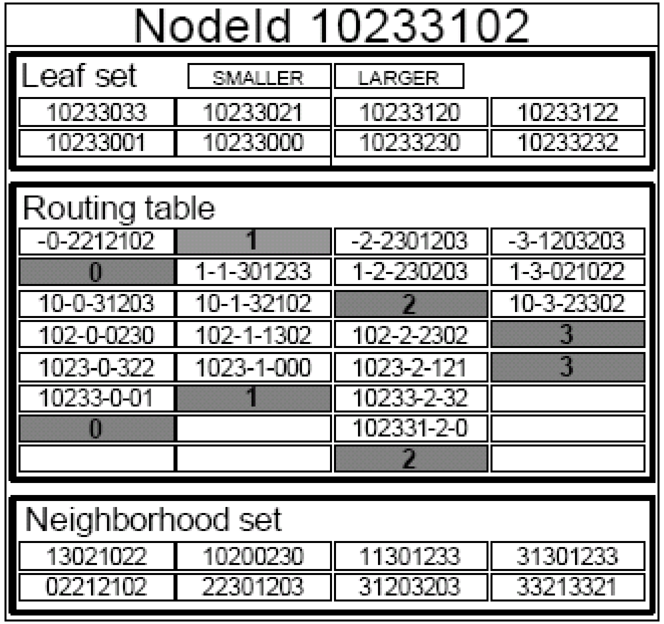
– Neighborhood set
  – Node IDs and IP addresses of the closest nodes according to proximity metric
  – not used for routing, but for self-organization

– Leaf set
  – Nodes closest in terms of node ID, |L|/2 nodes with higher and lower ID. L: constant parameter

3. Structured Peer-to-Peer Networks: DHTs

6.1 Pastry Routing Table

- b = 2, L=8
- All numbers in base 4
- Shaded cells show corresponding digits of present node ID.
- Notation: common digits - next digit - rest
- IP addresses are not shown.



NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

b UNIVERSITÄT BERN

3. Structured Peer-to-Peer Networks: DHTs

6.2 Pastry Routing

- Node ID of destination is compared to leaf set and message is delivered to destination node, if covered by leaf set.
- If not, routing table is used and message is forwarded to the node that shares a common prefix by at last one more digit.
- If this is not possible, message is forwarded to a node that
  - shares at least the same number of prefix digits as the local node and
  - is numerically closer than the local node.

UNIVERSITÄT BERN

3. Structured Peer-to-Peer Networks: DHTs

6.3 Pastry Self-Organization

Joining nodes

– Node X knows a nearby node A and sends a Join message to A.

– Node X asks A to route a Join message with the key equal to X.

– Routing of Join from A to Z, whose ID is numerically closest to X

– Nodes on the path deliver tables to X.

– X can request additional information and initializes tables.

Leaving nodes

– Nodes may leave without notification.

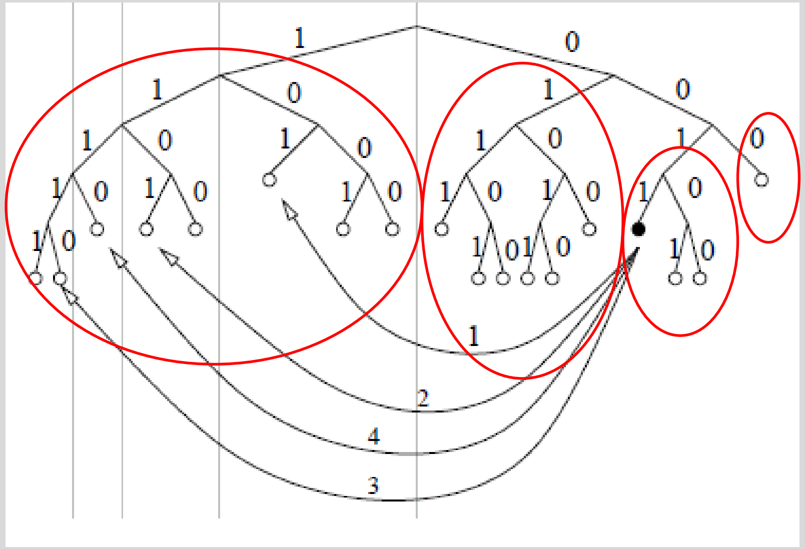
– Routing tables must be repaired.

– Nodes in the same row as the failed node can be asked.

3. Structured Peer-to-Peer Networks: DHTs

7. Kademlia

- 160 bit identifiers
- Storage of (key, value) pairs at nodes close to the key
- P2P systems based on Kademlia: eMule, eDonkey, BitTorrent
- Nodes = leaves in a binary tree
- for a given node: Tree is divided into subtrees, highest subtree consists of half of the tree not containing the node, next subtree consists of half of the remaining subtree not containing the node, …
- A node should know at least k=1 nodes in each subtree.
- Routing towards closer nodes with distance metric d(x,y) = x ⊕ y
- Example: Node 0011 finds node 1110.



Thanks
for Your Attention

Prof. Dr. Torsten Braun, Institut für Informatik

Bern, 28.09.2020