VU 185.A75 - 2018W
Peer-to-Peer Systems
Compact study guide

barfoos
< nopedope11 [at] gmail [dot] com >

1. Februar 2019

1 Introduction

1.1 What is a Pareto distribution, where is it applied to?

- Pareto-Dist: power-law probability distribution
- Related with power-law: proportional relationship between two quantities e.g. square side doubled, area is 4-times bigger
- Applied to e.g. distribution of wealth in society

1.2 How do join and query work in Gnutella?

- Gnutella: search protocol without central authority
- Messages: PING, QUERY (broadcast); PONG, QUERY HIT (back-propagated); PUSH (node-to-node)
- Join:
  1. New node creates TCP connection with known servent (server+client) in Gnutella
  2. Sends connection request "Gnutella Connect/4.0"
  3. Servent accepts with replying "Gnutella OK"
  4. Find other servents via ping/ping dialog
- Query (standard BFS):
  1. Node sends QUERY message to all neighbours
  2. They forward message to their neighbours (no-loop since saving query ID)
  3. Nodes that have the file initiate a QUERY HIT reply
  4. Query reply message is back-propagated
  5. File download occurs (via HTTP)

1.3 What are the properties of Gnutella Topology? What is small world property?

- Topology is a power-law graph with $N(d) = c \cdot d^{-r}$ number of nodes N with degree $0 < d \leq 7$, $avg(d) = 2.2$ and $r = 1.4$
- Central idea: "rich gets richer" as well-connected nodes attract more peers
- Experiment (Milgram)
  1. Copies of a letter (recv name & -address) distributed to random persons
  2. Only allowed to give them to friends
  3. Letter arrived after ≈ 6 hops

1.4 Comparison of node degree and network diameter in different p2p systems

	Gnutella	Chord	Kademlia
Node degree d	small $d = [8, 12]$	tiny $O(\log N) = 3$	constant
Network diameter D	small $D = [1.7, 4]$	tiny $O(\log N)$	tiny $O(\log N)$
Clustering coefficient c	high		

1.5 Indirection (de-referencing) principle in p2p networks

Reference something using e.g. a name instead of the value itself (most commonly through memory address)

(a) Space:

- Content-based IDs
- Routing to these (<> IP addressing)
- Routing even when nodes are joining/leaving the network

(b) Time:

- Send/receive are temporally independent: rendez-vous
- In the meantime: storage in the net (e.g. via TTL on storage)

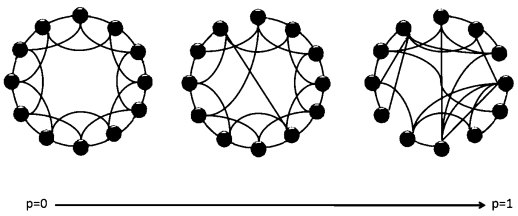
1.6 What is content-centric routing in p2p networks?

- Problem: transmission of large files (e.g. videos) through the network
- Solution: Routing paths are determined by content
- Advantages: improve power consumption, aggregated data transmits faster
- e.g. CLADA

1.7 How do Watts and Strogatz create an approximate small world topology?

Model of Watts/Strogatz ($\rightarrow$ diameter D increases)

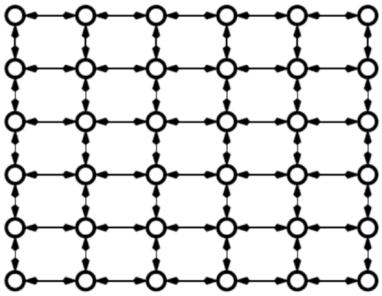
- Start with ring
- Each node linked to neighbours $k/2$ hops away
- Replace an original link with prob. p with link to random node



1.8 How does each node connect to the neighbours, according to Kleinberg, in order to achieve efficient routing?

Model of Kleinberg

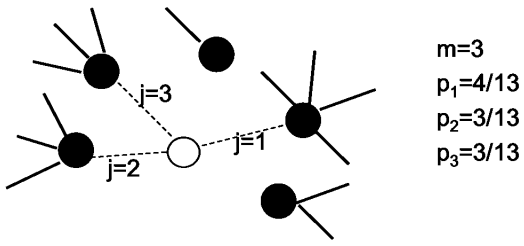
- Correlation between local and distant connections
- Nodes connected by the grid links
- Additionally two nodes u, v can be connected with prob. $[d(u,v)]^{-r}$
- Decentralized algorithm that constructs a path between any two nodes of length $O(\log N)$ only if $r = 2$.



1.9 How do the nodes connect according to Barabasi et al.?

Model of Barabesi/Albert (preferential attachment)

- New node(white) has preference to whom it wants to be connected
- Probability increases proportionally with increase of degree
- Model produces power law- or scale free network
- Approximates Gnutella best



1.10 What means a scale free network?

- Node degree distribution follows a power law (at least asymptotically)
- Is not dependent on number of Nodes N
- $P(d) \approx d^{-\gamma}$ for large ds with fixed parameter $2 < \gamma < 3$

1.11 What are the relationships between node degree, scale free, power law distribution, diameter, small world and Gnutella?

- Node degree d, diameter D with k hops, N nodes

$$d^k \leq N \qquad \leftrightarrow \qquad k \geq \frac{\log N}{\log d}$$

- Scale-free node degree topology is efficient $p(d) = c * d^{-r}$
- For Gnutella: $0 < d \leq 7$, $avg(d) = 2.2$, $var(d) = 1.63$

1.12 What are the caveats of Gnutella 0.4 and how was Gnutella improved?

- Inefficient, flooding creates high traffic load
- If file is more than H hops away, it is not found
- Resulting topology is random

1.13 Why did FastTrack have security problems?

- Architecture: node is either supernode or node (assigned to supernode)
- Each supernode has TCP connections with 30-50 supernodes
- Problems:
  - Used hashing algorithm UUHash (fast) that allows massive unnoticed corruption
  - Spread of corrupt and fake files in the network

2 Consistent Hashing, Chord

2.1 Chord node degree, diameter complexity

- Every node has m finger (node degree of m)
- The network diameter is $O(\log N)$ since the distance at every hop is halved
- Diameter D in worst case with high prob. $\lceil O(\log_2 N) \rceil$

A node's fingertable looks like the following (in general ith finger of p_x is $succ(x + 2^{i-1})$:

Index	Target ID	Successor
0	$P29 + 1 = P30$	$P39$
1	$P29 + 2 = P31$	$P39$
2	$P29 + 4 = P33$	$P39$
3	$P29 + 8 = P37$	$P39$
4	$P29 + 16 = P45$	$P68$
5	$P29 + 32 = P61$	$P84$
6	$P29 + 64 = P93$	$P106$

2.2 Node responsibility for regions in the ID-space

2.3 Routing

- How are node neighbours connected?
- How to find the responsible node?

2.4 How is locality handled?

2.5 Mechanisms for load balancing

- E.g. Different node performance

2.6 What is stored? Index or data?

- Each node stores local data
- Publishes whole data object XOR the index only
- Soft state update:
  - Destination node starts a timer after `publish`()
  - Requesting node has to re-publish before timeout
  - Node fails → the published data disappear after some time

2.7 Explain consistent Hashing

- Original idea: Distributing requests among a changing population of web caches
- Consistence: Ensure that add/remove of one cache does not significantly change the mapping of keys to caches
- e.g. insertion of peer p_5:

$$p_1(a,b,c), p_2(d,e,f), p_3(g,h,i), p_4(j,k) \longrightarrow p_1(a,b,c), p_2(d,e,f), p_5(g), p_3(h,i), p_4(j,k)$$

Theorem:

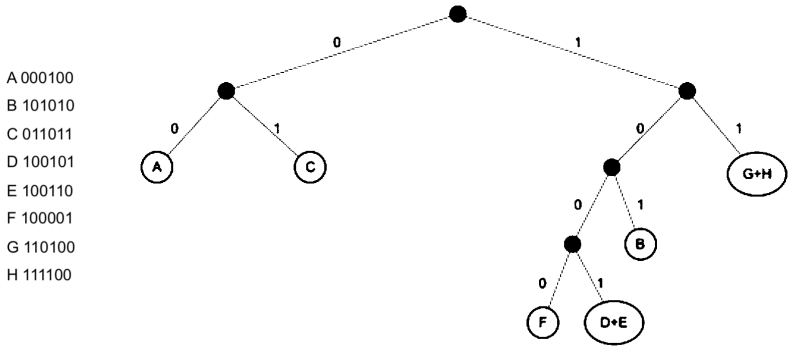
- Let N be the set of nodes and K the set of items
- It holds with high prob.: each node responsible for at most $\frac{K(1+\epsilon)}{N}$ items
- When a new peer $(N+1)$th joins the network, responsibility for $O(\frac{K}{N})$ peers changes.

3 Structured P2P networks CAN, P-Grid, Kademlia

3.1 How to build a P-Grid tree (given the data IDs) optimally with a certain number N of peers?

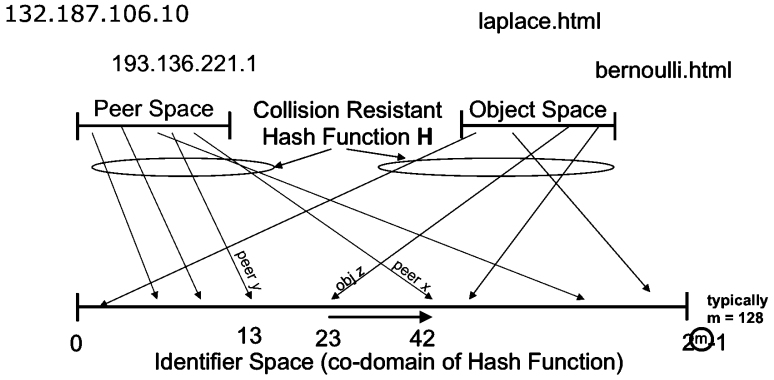
- While enough peers per data bucket, divide a data bucket in two halves (0 and 1 leaves)
- Routing performed on a binary tree (efficient structure)
- Each node has unique $0, 1$ sequence

e.g. with 6 peers:



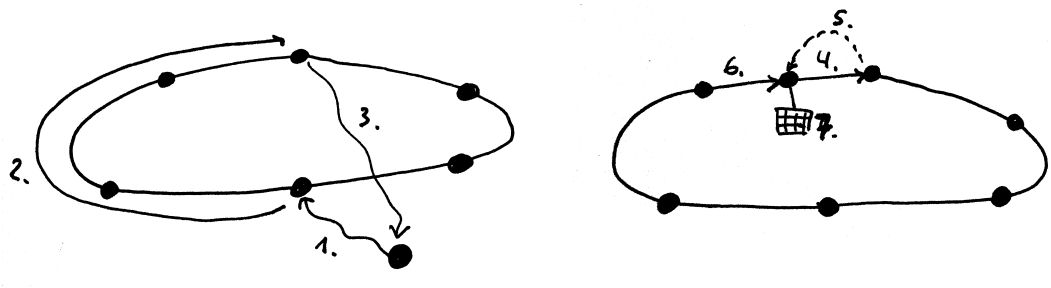
3.2 Chord: ID space, node responsibility, routing table

- ID space $[0, 2^m - 1]$, with typically $m = 128$
- Mapping `node_id` $= sha1($`node_ip`$)$



3.3 Explain the Chord join procedure

1. Send JOIN message to a (known) peer in the network
2. Route the query in the ring
3. Provide new peer's successor
4. Set successor
5. Redistribute routing information (of objects)
6. Update successor of predecessor
7. Build the fingers ($O(\log N)$ are built)



3.4 CAN

Routing hops $O(\sqrt{N})$, routing tables $O(1)$:

- ID space: d-dimensional torus ($d = 3$ is a 3D ring)
- Each node responsible for a rectangular zone
- Routing: select closest neighbour for next hop (smallest Euklidean distance)

$$\min \quad d(q,p) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Pos/Neg:

+ Simple mechanism
+ Small node degree
+ Robust network - neighbour links
– Long paths (polynomial)
– Few node neighbours (less stability)

3.5 CAN: Join

- New node n contacts a node in the CAN via bootstrap process
- n chooses point (x, y) in the space uniformly at random
- n informs node p that is responsible for (x, y) using CAN
- p splits zone in half ($1st$, then $2nd$, ... dimension, e.g. $x \to y \to z$)
- p keeps $1/2$ and n gets $1/2$
- n builds connections to it's neighbours

3.6 CAN: defragmentation

With CAN-zones as binary tree:

- Discover that the peer p failed through keep-alive messages
- Two cases:
  (a) Silbling is a leaf: smallest region is selected
  (b) Silbling is not a leaf: go to descendant (up the tree) until two leaves are found corresponding to two nodes

3.7 Kademlia

- ID space $[0, 2^m - 1]$, typically $m = 160$ bits
- Mapping `node_id` $= sha1($`node_ip`$)$
- Distance between nodes x, y: $d(x, y) = x \oplus y$
- Node with smallest (XOR)distance is responsible

3.8 Kademlia: Network structure

- For each of the $m = 160$ bits exists a list with max k elements (typically $k = 20$) called a k-bucket
- Candidates for a k-bucket are selected by their online time (decreasing, first k are selected)
- Each node has their own k-buckets, up to $k \cdot m$ entries of form $\langle \text{IP}, \text{UDP port}, \text{node ID} \rangle$

3.9 Kademlia: Bucket building at a new node

$m = 4, k = 2$ known peer b and new node n:

- Build bucket scheme for n according to m to cover all other nodes in the form e.g. $1xxx$, $01xx$, $001x$, 0000 iff $n = 0001$
- n contacts b and ask for k neighbours of it
- Refresh procedure: repeat with found nodes until all buckets have elements

4 Pastry & Tapestry

4.1 Pastry

- ID space $[0, 2^{b \cdot l} - 1]$, typically $m = b \cdot l = 128$ bits
- Mapping `node_id` $= sha1($`node_ip`$)$
- `node_id` is a number of 2^b letters with length $l = 32$ (typically)

4.2 Pastry heuristic

- If target is within range of leaf set $\rightarrow$ forward to numerically closest peer
- Else if R_l^d exists, forward to it with length of shared prefix l, $d = D[l]$
- Else forward to known node and shares a long prefix and is numerically closer than this node

4.3 Pastry prefix routing, route length, table size

- Prefix routing: send the message from a (using as routing table) to the node b with longest common prefix (if the target is in as leaf set, the message is routed directly to b)
- If there is no matching node, the cell is empty
- Route length: $O(\log_2^b N)$ hops

Routing table with m/b rows and 2^b columns:

	0	1	2	3
0	-0-3323	-1-3211	-2-3130	-3-0120
1	2-0-310	–	2-2-220	2-3-130
2	23-0-01	23-1-30	–	–
3	–	–	–	–
4	–	–	–	–

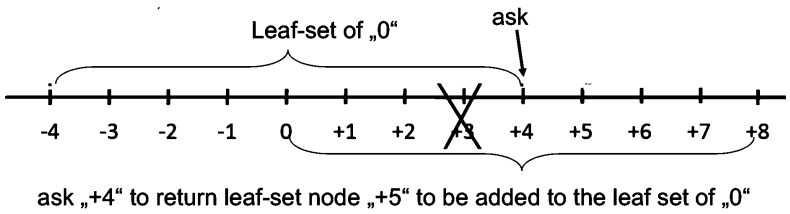
4.4 Pastry node failure

Periodical keep-alive messages:

(a) Eager leaf set repair:
- All the time
- Dead node → ask furthest alive node of the leaf set for his leaf set

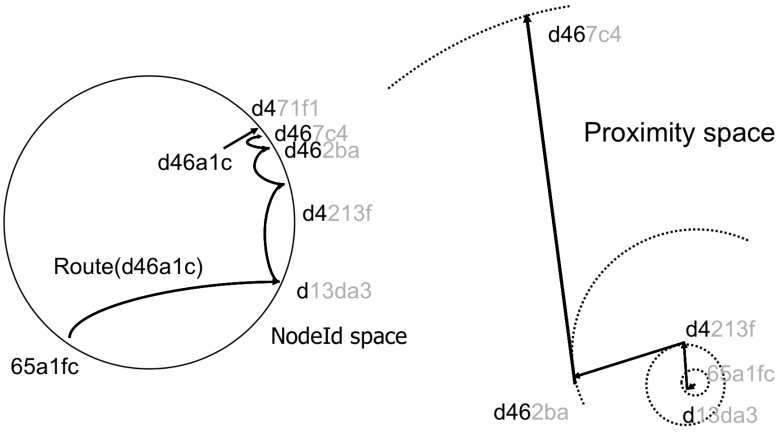
(b) Lazy routing table repair:
- Get table from peers in the same row
- If not found from higher rows

(c) Neighbourhood set repair:
- All the time



4.5 Locality properties

- Neighbourhood in the hash space <> geographic proximity
- Proximity metric: probe the distance to another node via ping/RTT (round trip time delay)/number of IP hops
- Proximity invariant: in the proximity space should contain the nearest nodes, highest level = nearest nodes (because of repair), further levels = further distances



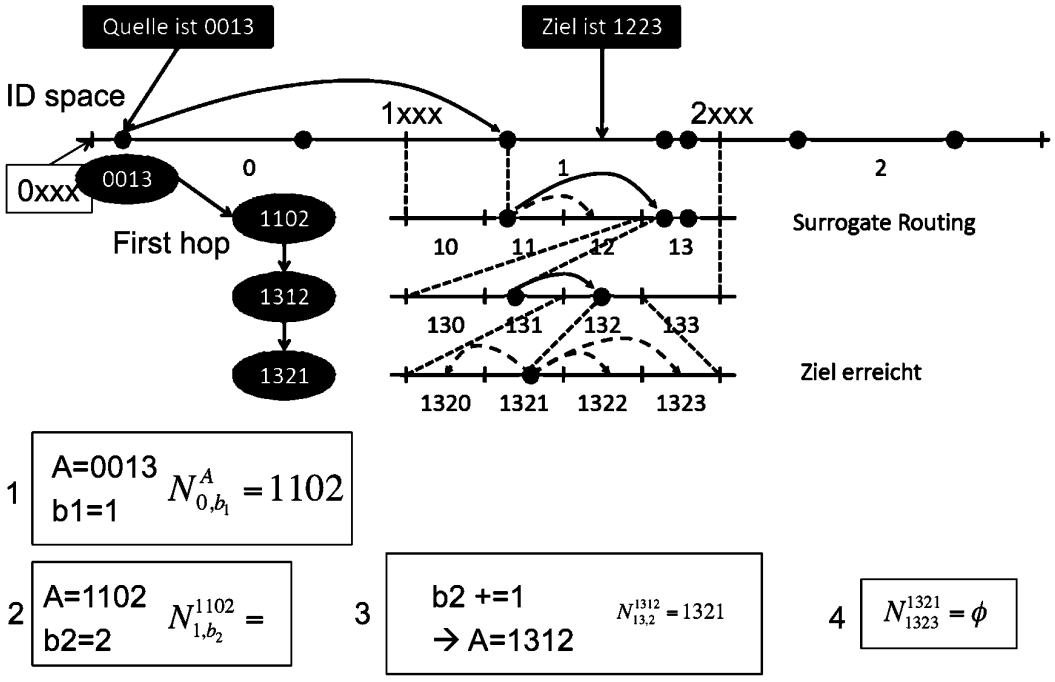
4.6 Replication in pastry

- DHT entries replicated on other nodes (increase availability in case the node fails)
- r-replicas with system parameter r and $r < L$
- Replica invariant: Pastry maintains r replicas of the closest (numerically, ID) independent of node insertion/failures

4.7 Tapestry: explain the routing idea (surrogate routing)

Find the root peer of an ID:

- Start from level 1 to route to primary neighbours, increase level
- If (x, j) neighbour is empty, increase $j\%B$ until a node is found
- If no node is found, the current node is the root-peer (destination)
- Done



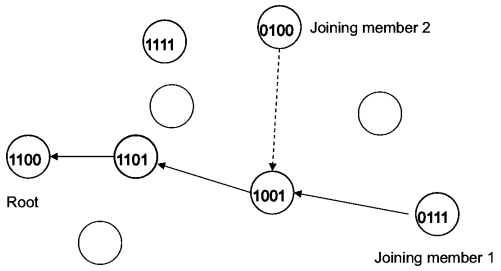
5 Application layer Multicast, BitTorrent

5.1 Scribe: functionality (API)

- Start from level 1 to route to primary neighbours, increase level
- If (x, j) neighbour is empty, increase $j\%B$ until a node is found
- If no node is found, the current node is the root-peer (destination)
- Done

5.2 Scribe: Join

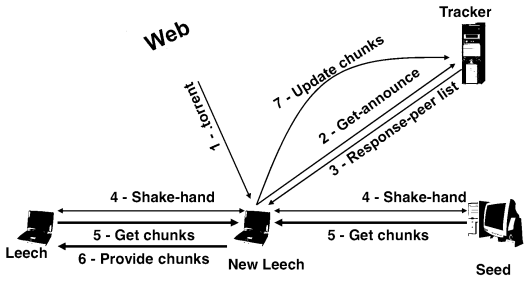
- Send JOIN message to groupID
- Each node on the path to groupID checks if it is already a "forwarder" of that group
- yes: accept the preceeding node as a child, terminates forwarding the message
- no: create an entry for group, add the preceeding node as child, forwards the message



5.3 Scribe: Multicast

5.4 BitTorrent: describe the phases (protocol) of BT

1. Get torrent (filename, size, checksum, tracker) from web
2. Contact tracker: maintains list of active peers, 1 tracker and 1 network (swarm) per file
3. Response peer list: ≤ 50 peers (seeders + leechers), seeders have the whole file, leechers only parts
4. Shake-hand: establish connection to 20-30 peers (prioritize by their bandwidth periodically)
5. Get chunk: request chunk (prioritize by rarity), download concurrently
6. Provide chunk: tit-for-tat strategy peers give as long as they receive, slow when no chunks available
7. Send update information to tracker



5.5 BitTorrent Pros/Cons

+ Good bandwidth utilization
+ Limits free riding (tit for tat)
+ Limits leech attack (couples download and upload)
+ Spurious (fake) files are not propagated
+ Open source implementations
+ Preferred for legal content distribution (updates, etc.)

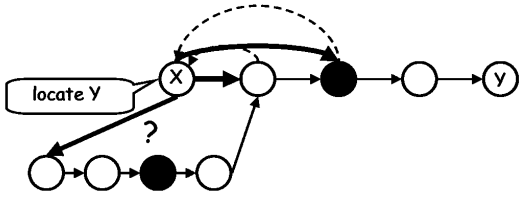
– Small files (latency, overhead)
– Central tracker server to bootstrap swarm (single point of failure, scalability)
– Robustness (progress depends on user behaviour)
– Centralized supervision of trackers

5.6 How would an adversary organize a Sybil attack on a DHT net?

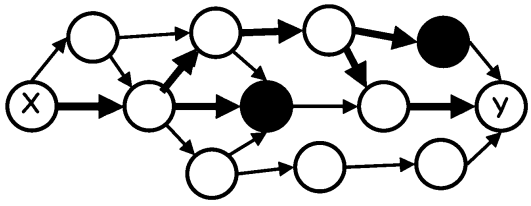
- Generate a lot of identities
- Goal: get into everyone's DHT routing table
- Reply with invalid data
- Introduce delays and outages
- Long-term: 80% sybils replying with invalid info (=censor honest peers)

5.7 Security: Iterative and disjoint path routing mechanisms

- Iterative:
  - Source node controls each routing step
  - Verify if the message moves with each operation comes closer to destination



- Disjoint:
  - Perform α parallel queries
  - Take k closest known nodes
  - Distribute them (uniformly) among the α search process
  - Each of the α lookups are handled like normal Kademlia lookups
  - Idea: Adversarial nodes cannot influence all search processes



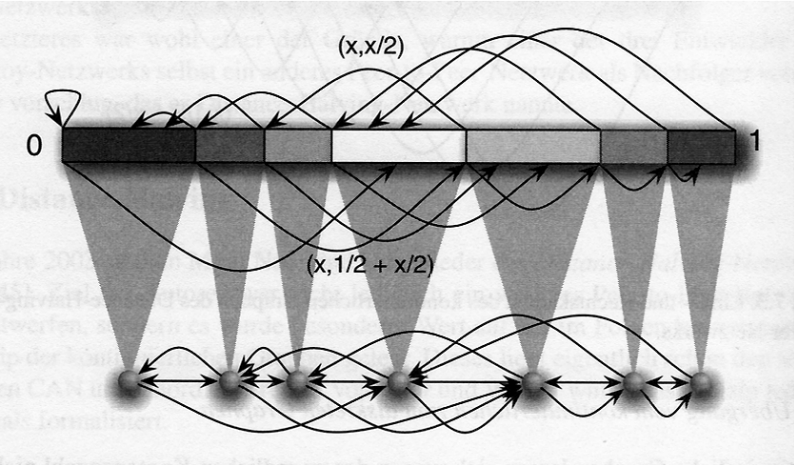
6 Constant degree p2p networks

6.1 Definition of continuous distance halving network

- Continuous Interval $[0, 1)$
- Set of nodes $V = [0, 1)$
- A set of arcs $E \subset V \times V$
- For every node $x \in [0, 1)$:
  - left-arc $(x, \frac{x}{2})$
  - right-arc $(x, \frac{x+1}{2})$
  - reverse left arc $(\frac{x}{2}, x)$
  - reverse right arc $(\frac{1}{2} + x, x)$

6.2 Definition of discrete distance halving network

- Created from continuous graph
- n points with $\forall i \in \mathbb{N} : x_i \in [0, 1]$
- Segment maps to a node with $s(x_i) = [x_i, x_{i+1}]$
- Edge between $s(x_i)$ and $s(x_j)$ exists iff two points $u \in s(x_i)$ and $v \in s(x_j)$ such that the arc (u, v) exists in the continuous graph



6.3 Explain why is the DH node degree constant

- Principle of multiple choice:
  1. Assuming there are nodes present in the discrete DH network
  2. Goal: insert next node at best position
  3. Make $k = c \log n$ trials at random positions $\in (0, 1)$ and choose the position belonging to the largest segment

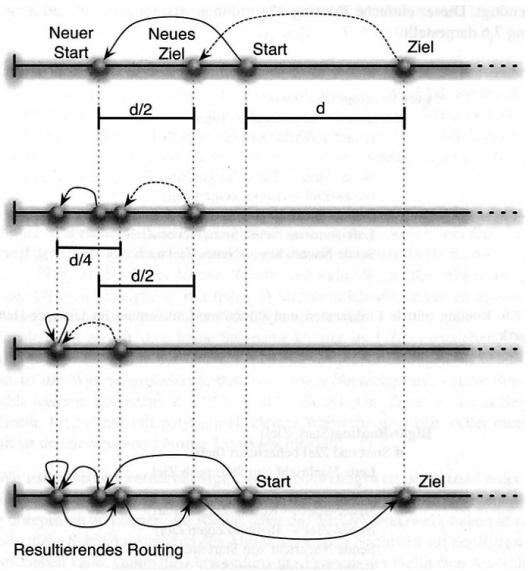
4. Divide that segment in the middle

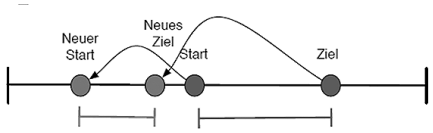
- Insertion of 2^k nodes with the MC-principle will result in sizes of $\frac{1}{2n}$, $\frac{1}{n}$ and $\frac{2}{n}$
- Therefore left/right arcs can be drawn to only 2 intervals
- Therefore the node degree is constant

6.4 Routing idea in DH, path length

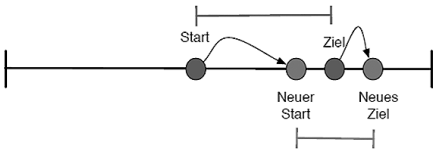
Start x, target y, distance $d = |x - y|$

- Use left/right arcs from start and target
- At each step $d_{i+1} = d_i/2$
- Path length: $2 \log n + 3$





- Using left –arc (the distance start-dest halves)



- Using right-arcs (the distance start-dest halves as well)

6.5 De Brujin network

- Initial k-string S_i can reach any target string S_f through sequences of shuffle `S` and shuffle-exchange `SE` operations
- Magic trick: $S_i \oplus S_f$ with $1 = \texttt{SE}$ and $0 = \texttt{S}$
- Can easily translated to a graph with 2^k nodes

Properties:

- k nodes, 2^{k+1} edges
- Out-degree 2
- In-degree 2 for `SE` node
- Diameter $D = k$

$$\begin{aligned} succ(x) &= 2x \ \% \ 2^m \\ &= 2x + 1 \ \% \ 2^m \qquad \text{(beides)} \end{aligned}$$

6.6 Routing in Koorde

(a) Relationship on the ring

(b) DeBruijn edges

- $2x$ and $2x + 1$ edges point to the same real node
- Use only one edge to node $2x$

7 Security issues in p2p

7.1 What is a sybil attack?

- Malicious node that creates multiple virtual nodes with different IDs that participate to the network
- Occupy certain ID (hash) domain
- Fill in the routing table of the attacked node with own entries
- Control the outgoing messages of the attacked node (eclipse attack)

Goals:

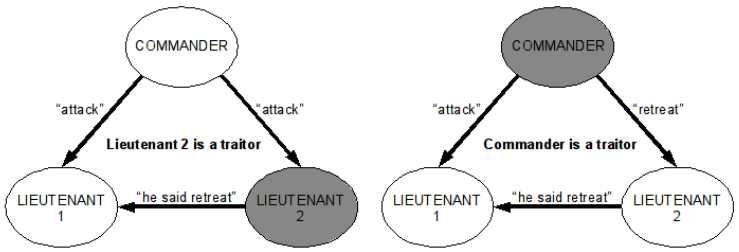
- Enables other attacks (e.g. route table poisoning)
- Attack connectivity of p2p network (fail of malicious node → honest nodes fail as well)
- Undermine data replication
- Disrupt majority votes (e.g. join permission)

7.2 Routing table poisoning

- Attacker forwards messages incorrectly or sends incorrect routing updates
- Attacker answers only with malicious nodes
- Bad nodes dominate over time

7.3 Byzantine generals (majority decision)

- Problem with finding consensus in presence of traitors
- Generals have to vote $v \in \{attack, retreat\}$
- $3m + 1$ generals for m traitors for majority consensus



7.4 What are eclipse attacks and how do they work?

- Malicious nodes try to dominate the neighbour set of honest nodes
- Goal: dominate the network's traffic, eclipse honest nodes from each other's view
- Chord: by algorithm (not vulnerable), Gnutella: neighbours freely selectable (vulnerable)

Attack

- Sybil attack to increase number of malicious nodes
- Insert malicious nodes into the routing tables via neighbour discovery and routing table maintenance
- Pretend to be close to the attacked node (proximity selection)

Defend

- Observe: A node of an attacker in an Eclipse attack has significantly more input links (well-connected) than normal nodes
- Fight the Sybil attack
- k-buckets with update strategy like in Kademlia a→ good old nodes stay in the neighbourhood
- Limit the number of links

7.5 Poisoning attacks

Attackers use false information to break the integrity of the system

- Index Poisoning
  - Redirect requests to nodes that do not have the requested item
  - Redirect requests to attacker nodes
- File Poisoning
  - Spamming the network with fake and corrupted files
- Routing Table poisoning: routing attacks