# Lectures in Wroclaw

- **Epidemic Algorithms**
  - Monday, April 6th, 2009, 3pm
- **Random Networks**
  - Monday, April 6th, 2009, 6pm
- **Distributed Heterogeneous Hash Tables**
  - Tuesday, April 7th, 2009, 3pm
- **Network Coding**
  - Wednesday, April 8th, 2009, 11am
- **Locality in Peer-to-Peer Networks**
  - Wednesday, April 8th, 2009, 3pm



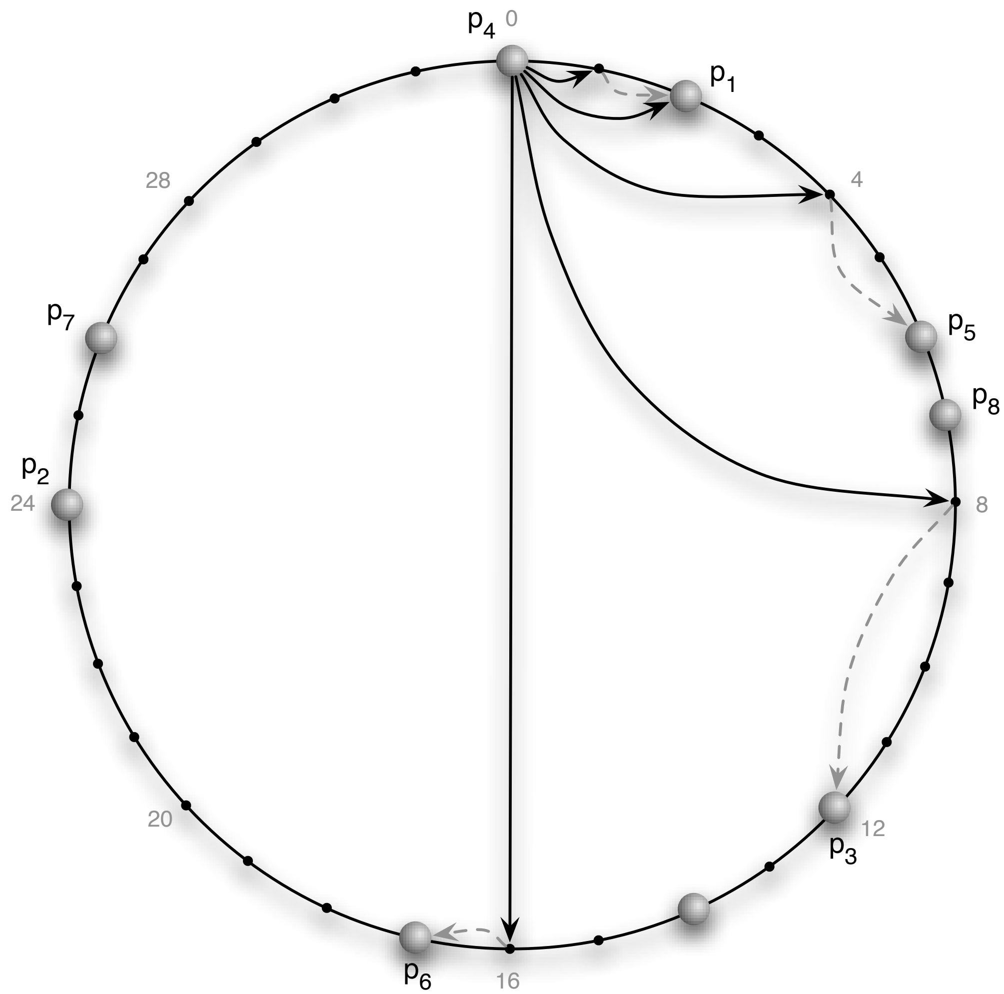
Computer Networks and Telematics
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

Peer-to-Peer Networks

# Chord

# Pointer Structure of  Chord

- **For each peer**
  - successor link on the ring
  - predecessor link on the ring
  - for all i ∈ {0,..,m-1}
    - Finger[i] := the peer following the value $r_V(b+2^i)$
- **For small i the finger entries are the same**
  - store only different entries
- **Lemma**
  - The number of different finger entries is O(log n) with high probability, i.e. $1-n^{-c}$.



# Properties of the DHT

- **Lemma**
  - For all peers b the distance $|r_V(b.succ) - r_V(b)|$ is
    - in the expectation $2^m/n$,
    - $O((2^m/n) \log n)$ with high probability (w.h.p.)
    - $2^m/n^{c+1}$ für a constant $c>0$ with high probability
  - In an interval of length $w\, 2^m/n$ we find
    - $\Theta(w)$ peers, if $w=\Omega(\log n)$, w.h.p.
    - at most $O(w \log n)$ peers, if $w=O(\log n)$, w.h.p.
- **Lemma**
  - The number of nodes who have a pointer to a peer b is $O(\log^2 n)$ w.h.p.

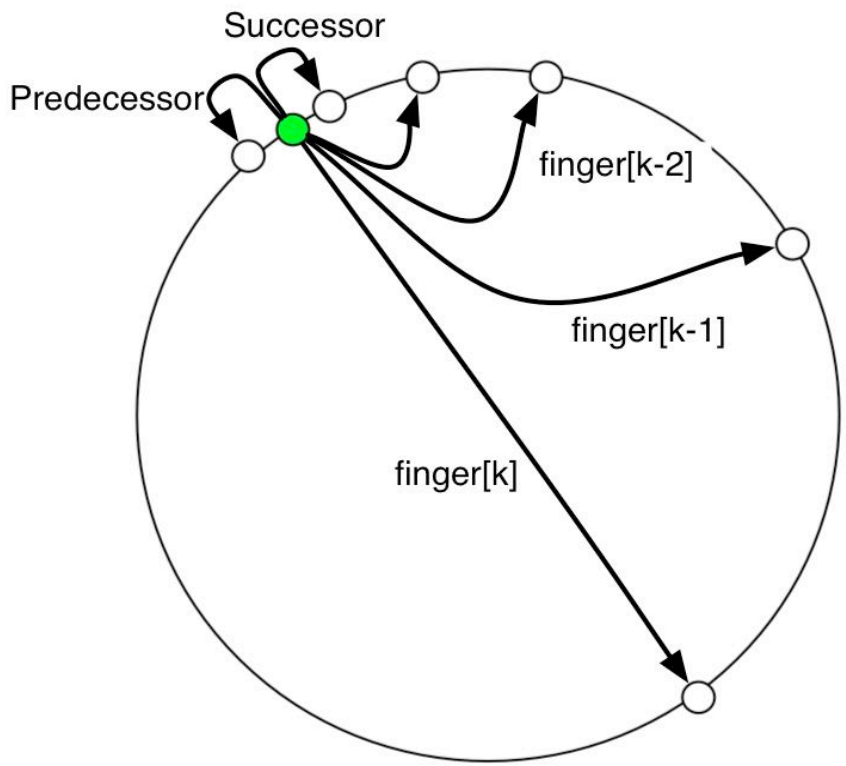
# Lookup in Chord

- **Theorem**
  - The Lookup in Chord needs O(log n) steps w.h.p.
- **Lookup for element s**
  - Termination(b,s):
    - if peer b,b'=b.succ is found with $r_K(s) \in [r_V(b), r_V(b'))$
  - **Routing**:
    Start with any peer b

```
while not Termination(b,s) do
    for i=m downto 0 do
        if r_K(s) ∈ [r_V(b.finger[i]),r_V(finger[i+1])] then
            b ← b.finger[i]
        fi
    od
```

# Data Structure of Chord

- **For each peer**
  - successor link on the ring
  - predecessor link on the ring
  - for all i ∈ {0,..,m-1}
    - Finger[i] := the peer following the value $r_V(b+2^i)$
- **For small i the finger entries are the same**
  - store only different entries
- **Chord**
  - needs O(log n) hops for lookup
  - needs O($\log^2$ n) messages for inserting and erasing of peers



Peer-to-Peer Networks

# DHash++

# Routing-Techniques for CHORD: DHash++

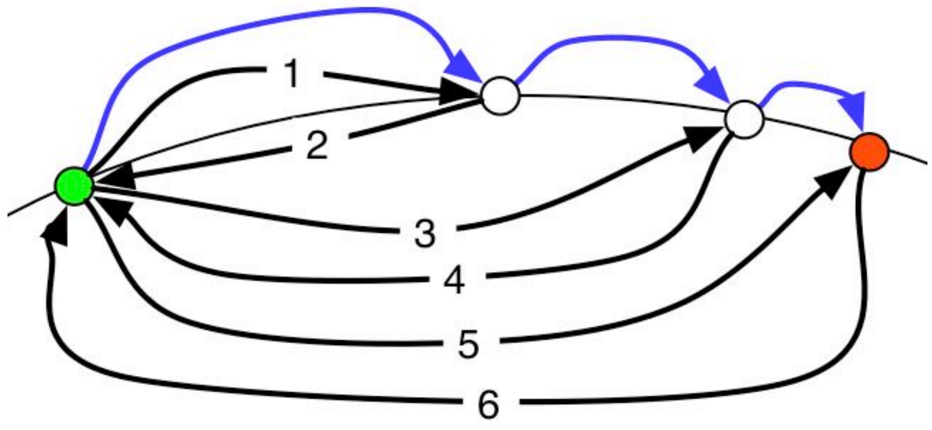
- **Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, Robert Morris (MIT)**
  **„Designing a DHT for low latency and high throughput“, 2003**
- **Idea**
  - Take CHORD
- **Improve Routing using**
  - Data layout
  - Recursion (instead of Iteration)
  - Next Neighbor-Election
  - Replication versus Coding of Data
  - Error correcting optimized lookup
- **Modify transport protocol**

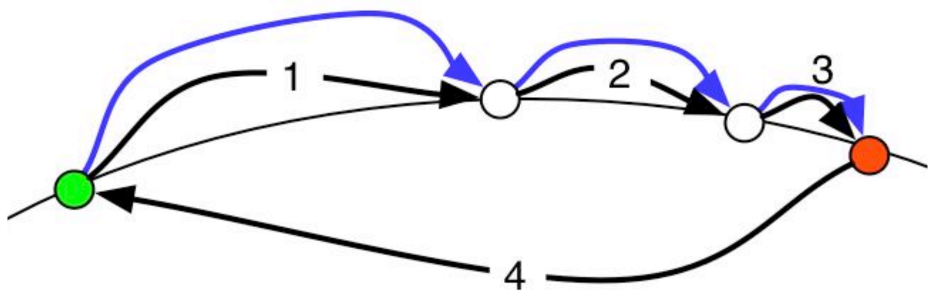
# Data Layout

- **Distribute Data?**
- **Alternatives**
  - Key location service
    - store only reference information
  - Distributed data storage
    - distribute files on peers
  - Distributed block-wise storage
    - either caching of data blacks
    - or block-wise storage of all data over the network

# Recursive Versus Iterative Lookup

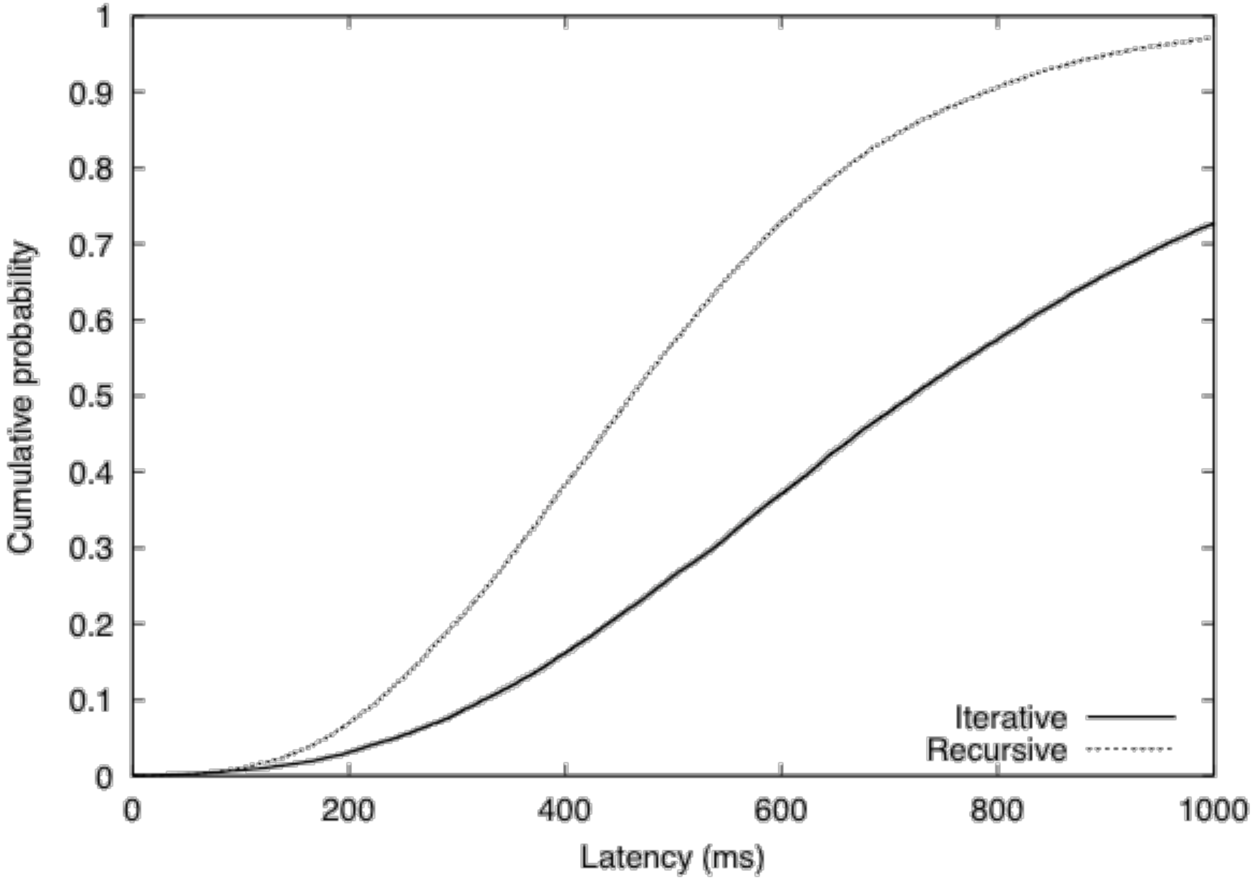
- **Iterative lookup**
  - Lookup peer performs search on his own
- **Recursive lookup**
  - Every peer forwards the lookup request
  - The target peer answers the lookup-initiator directly
- **DHash++ choses recursive lookup**
  - speedup by factor of 2



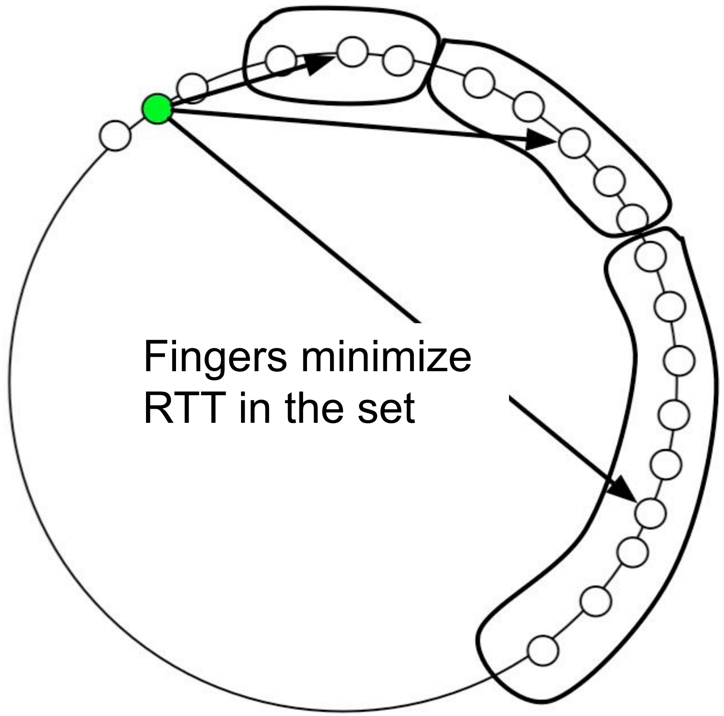
# Recursive Versus Iterative Lookup

- **DHash++ choses recursive lookup**
  - speedup by factor of 2



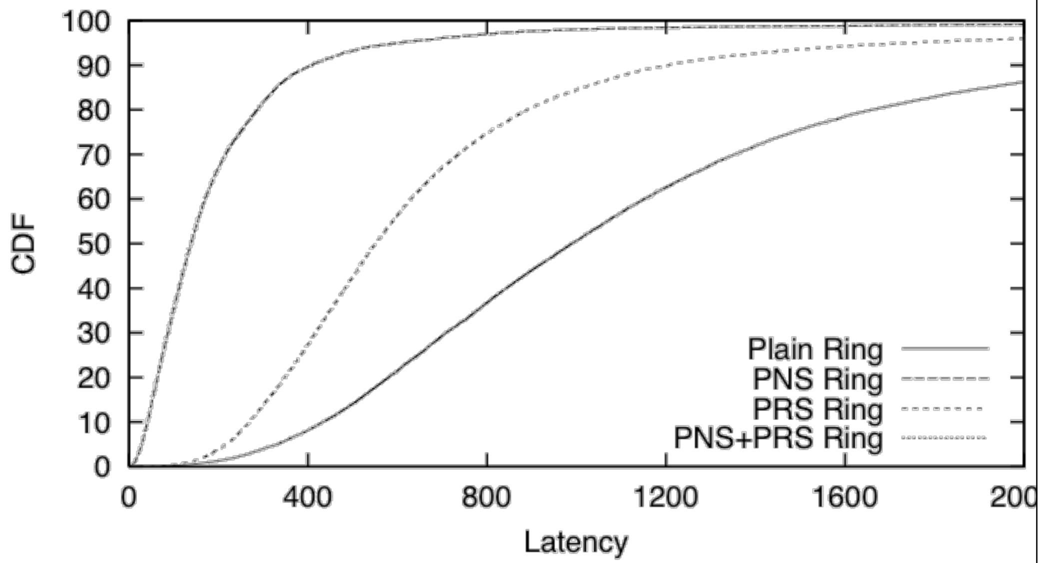
# Next Neighbor Selection

- **RTT: Round Trip Time**
  - time to send a message and receive the acknowledgment
- **Method of Gummadi, Gummadi, Grippe, Ratnasamy, Shenker, Stoica, 2003, „The impact of DHT routing geometry on resilience and proximity“**
  - Proximity Neighbor Selection (PNS)
    - Optimize routing table (finger set) with respect to (RTT)
    - method of choice for DHASH++
  - Proximity Route Selection(PRS)
    - Do not optimize routing table choose nearest neighbor from routing table

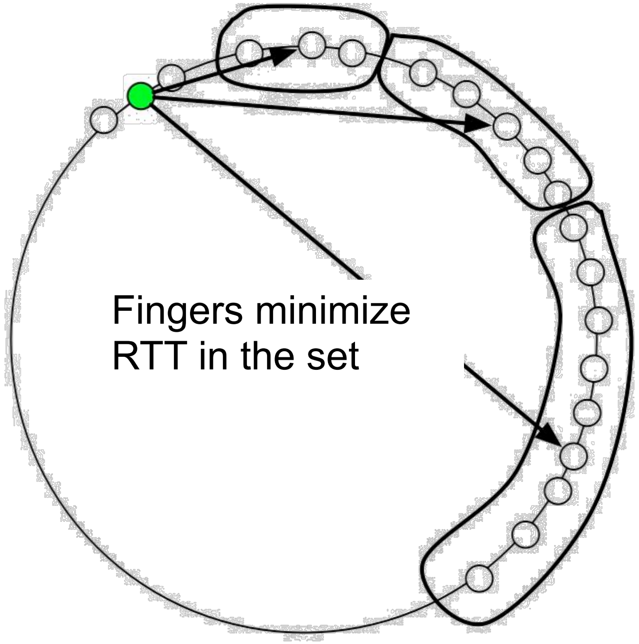


# Next Neighbor Selection

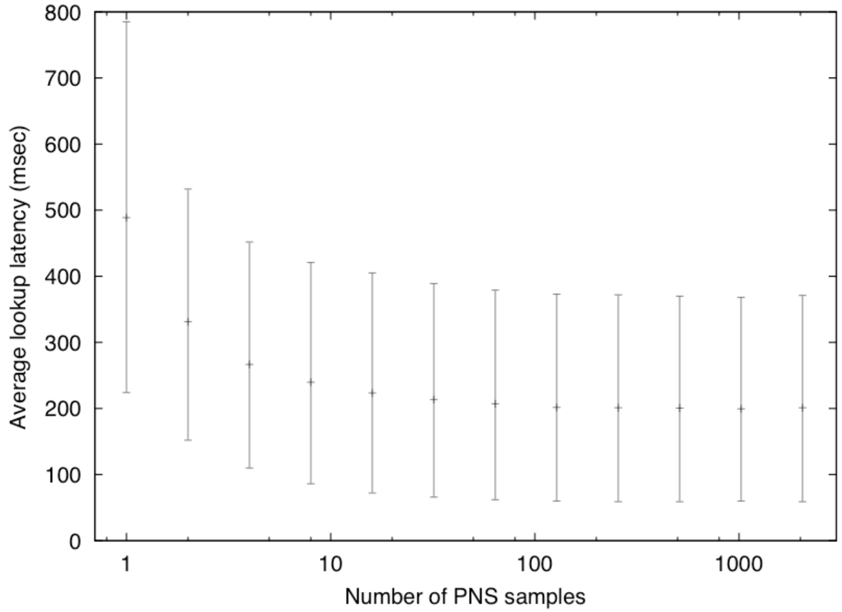
- **Gummadi, Gummadi, Grippe, Ratnasamy, Shenker, Stoica, 2003, „The impact of DHT routing geometry on resilience and proximity"**
  - Proximity Neighbor Selection (PNS)
    - Optimize routing table (finger set) with respect to (RTT)
    - method of choice for DHASH++
  - Proximity Route Selection(PRS)
    - Do not optimize routing table choose nearest neighbor from routing table
- **Simulation of PNS, PRS, and both**
  - PNS as good as PNS+PRS
  - PNS outperforms PRS



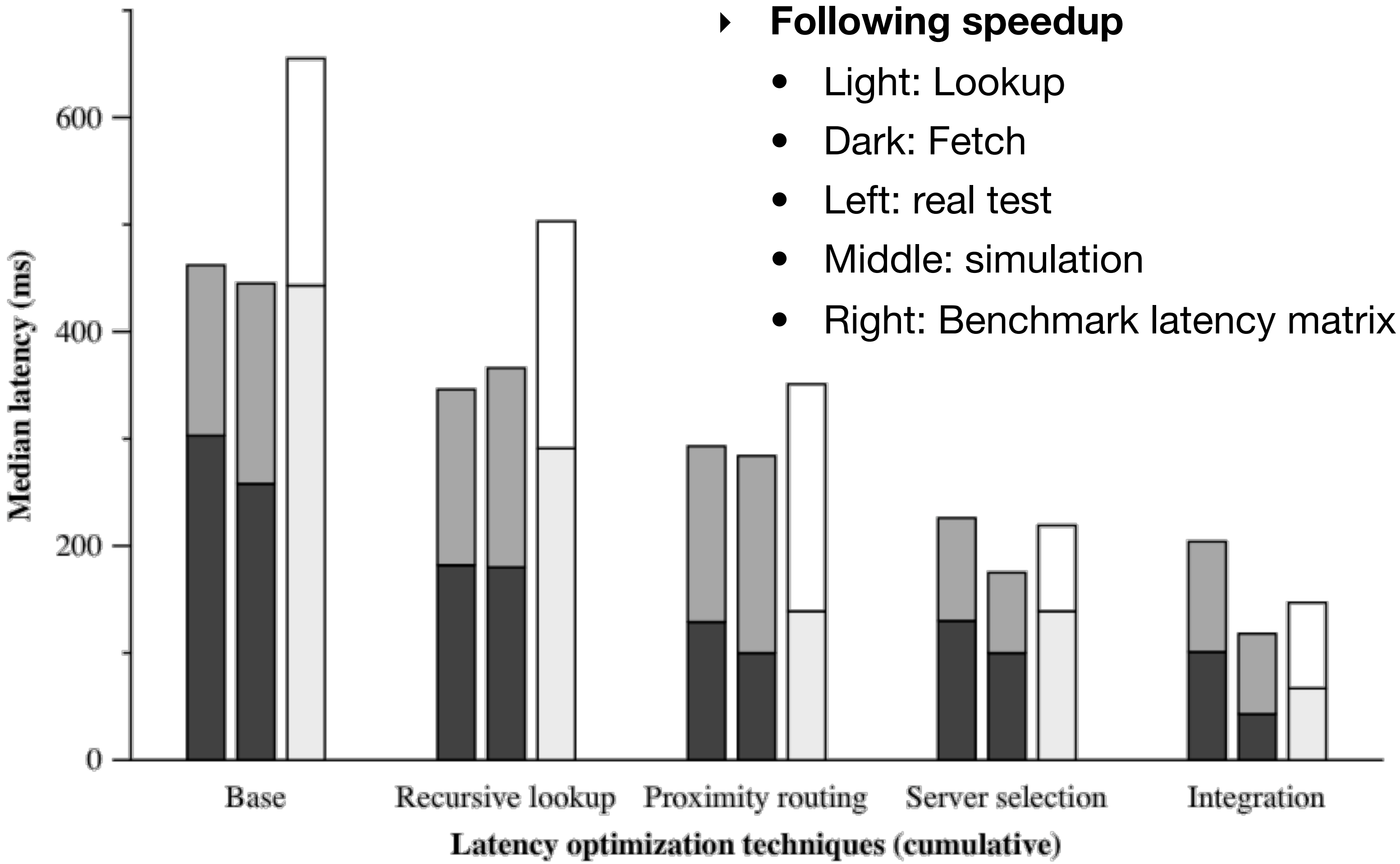
# Next Neighbor Selection




- **DHash++ uses (only) PNS**
  - Proximity Neighbor Selection
- **It does not search the whole interval for the best candidate**
  - DHash++ chooses the best of 16 random samples (PNS-Sample)
- **The right figure shoes the (0.1,0.5,0.9)-percentile of such a PNS-Sampling**

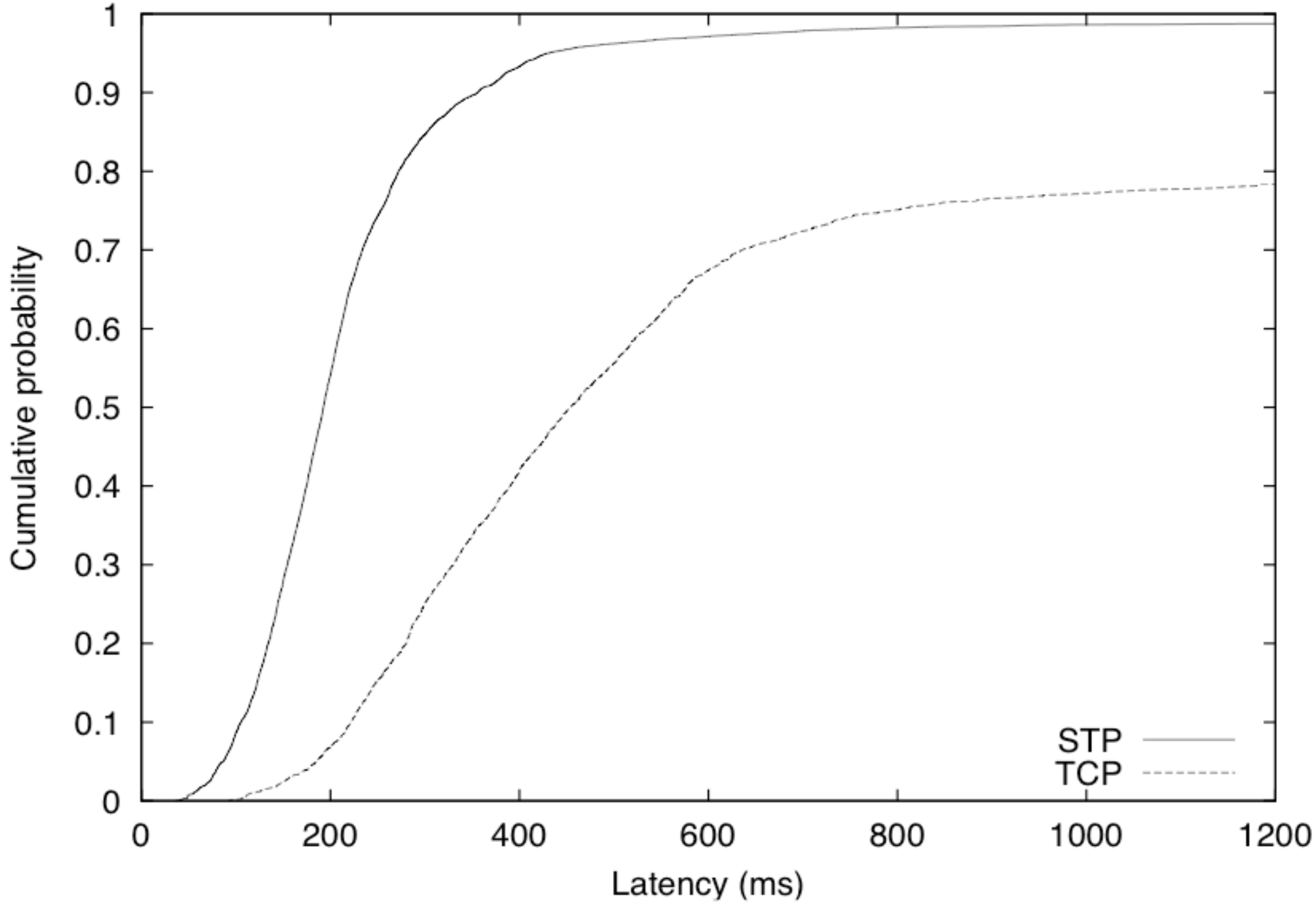


# Cumulative Performance Win




- **Following speedup**
  - Light: Lookup
  - Dark: Fetch
  - Left: real test
  - Middle: simulation
  - Right: Benchmark latency matrix

# Modified Transport Protocol



# Discussion DHash++

- **Combines a large quantity of techniques**
  - for reducing the latecy of routing
  - for improving the reliability of data access
- **Topics**
  - latency optimized routing tables
  - redundant data encoding
  - improved lookup
  - transport layer
  - integration of components
- **All these components can be applied to other networks**
  - some of them were used before in others
  - e.g. data encoding in Oceanstore
- **DHash++ is an example of one of the most advanced peer-to-peer networks**

## Peer-to-Peer Networks

# Pastry

# Pastry

- **Peter Druschel**
  - Rice University, Houston, Texas
  - now head of Max-Planck-Institute for Computer Science, Saarbrücken/ Kaiserslautern
- **Antony Rowstron**
  - Microsoft Research, Cambridge, GB
- **Developed in Cambridge (Microsoft Research)**
- **Pastry**
  - Scalable, decentralized object location and routing for large scale peer-to-peer-network
- **PAST**
  - A large-scale, persistent peer-to-peer storage utility
- **Two names one P2P network**
  - PAST is an application for Pastry enabling the full P2P data storage functionality
  - First, we concentrate on Pastry

# Pastry Overview

- **Each peer has a 128-bit ID: nodeID**
  - unique and uniformly distributed
  - e.g. use cryptographic function applied to IP-address
- **Routing**
  - Keys are matched to $\{0,1\}^{128}$
  - According to a metric messages are distributed to the neighbor next to the target
- **Routing table has $O(2^b (\log n)/b) + \ell$ entries**
  - n: number of peers
  - $\ell$: configuration parameter
  - b: word length
    - typical: b= 4 (base 16), $\ell = 16$
    - message delivery is guaranteed as long as less than $\ell/2$ neighbored peers fail
- **Inserting a peer and finding a key needs $O((\log n)/b)$ messages**

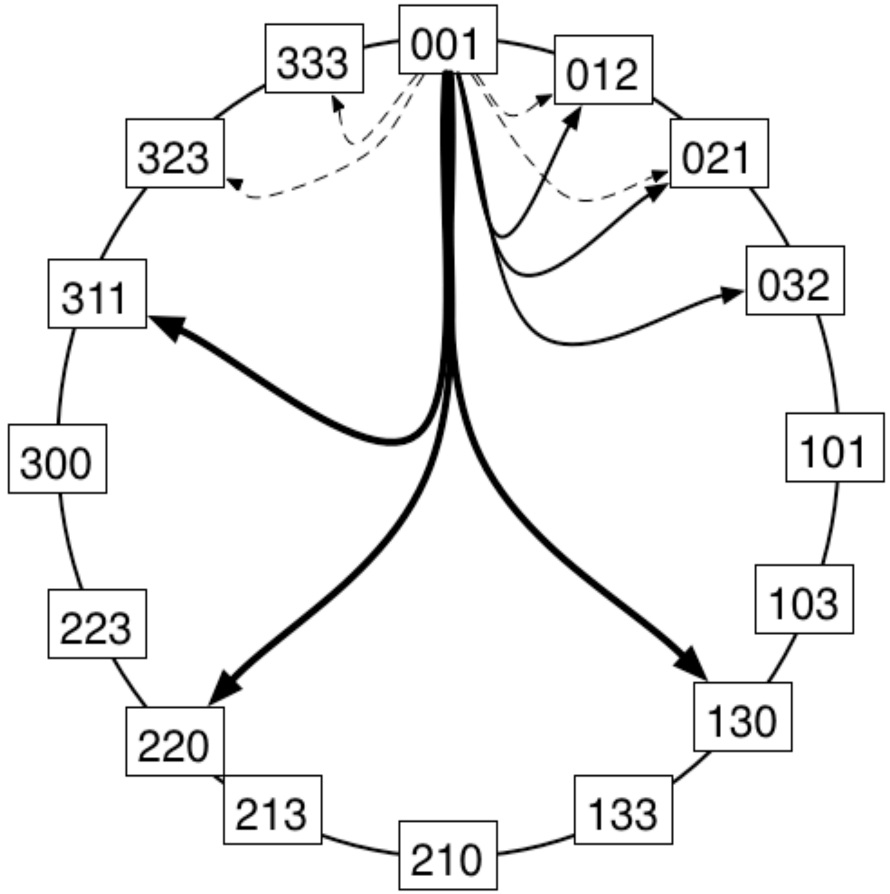
# Routing Table

- **NodeId presented in base $2^b$**
  - e.g. NodeID: 65A0BA13
- **For each prefix p and letter x ∈ {0,..,$2^b$-1} add an peer of form px* to the routing table of NodeID, e.g.**
  - b=4, $2^b$=16
  - 15 entries for 0*,1*, .. F*
  - 15 entries for 60*, 61*,... 6F*
  - ...
  - if no peer of the form exists, then the entry remains empty
- **Choose next neighbor according to a distance metric**
  - metric results from the RTT (round trip time)
- **In addition choose $\ell$ neighbors**
  - $\ell$/2 with next higher ID
  - $\ell$/2 with next lower ID

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | 1x | 2x | 3x | 4x | 5x | | 7x | 8x | 9x | ax | bx | cx | dx | ex | fx |
| 60x | 61x | 62x | 63x | 64x | | 66x | 67x | 68x | 69x | 6ax | 6bx | 6cx | 6dx | 6ex | 6fx |
| 650x | 651x | 652x | 653x | 654x | 655x | 656x | 657x | 658x | 659x | | 65bx | 65cx | 65dx | 65ex | 65fx |
| 65a0x | | 65a2x | 65a3x | 65a4x | 65a5x | 65a6x | 65a7x | 65a8x | 65a9x | 65aax | 65abx | 65acx | 65adx | 65aex | 65afx |

# Routing Table

- **Example b=2**
- **Routing Table**
  - For each prefix p and letter x ∈ {0,..,$2^b$-1} add an peer of form px* to the routing table of NodeID
- **In addition choose ℓ neighors**
  - ℓ/2 with next higher ID
  - ℓ/2 with next lower ID
- **Observation**
  - The leaf-set alone can be used to find a target
- **Theorem**
  - With high probability there are at most O($2^b$ (log n)/b) entries in each routing table



# Routing Table

- **Theorem**
  - With high probability there are at most $O(2^b (\log n)/b)$ entries in each routing table
- **Proof**
  - The probability that a peer gets the same m-digit prefix is

$$2^{-bm}$$

  - The probability that a m-digit prefix is unused is

$$(1 - 2^{-bm})^n \leq e^{-n/2^{bm}}$$

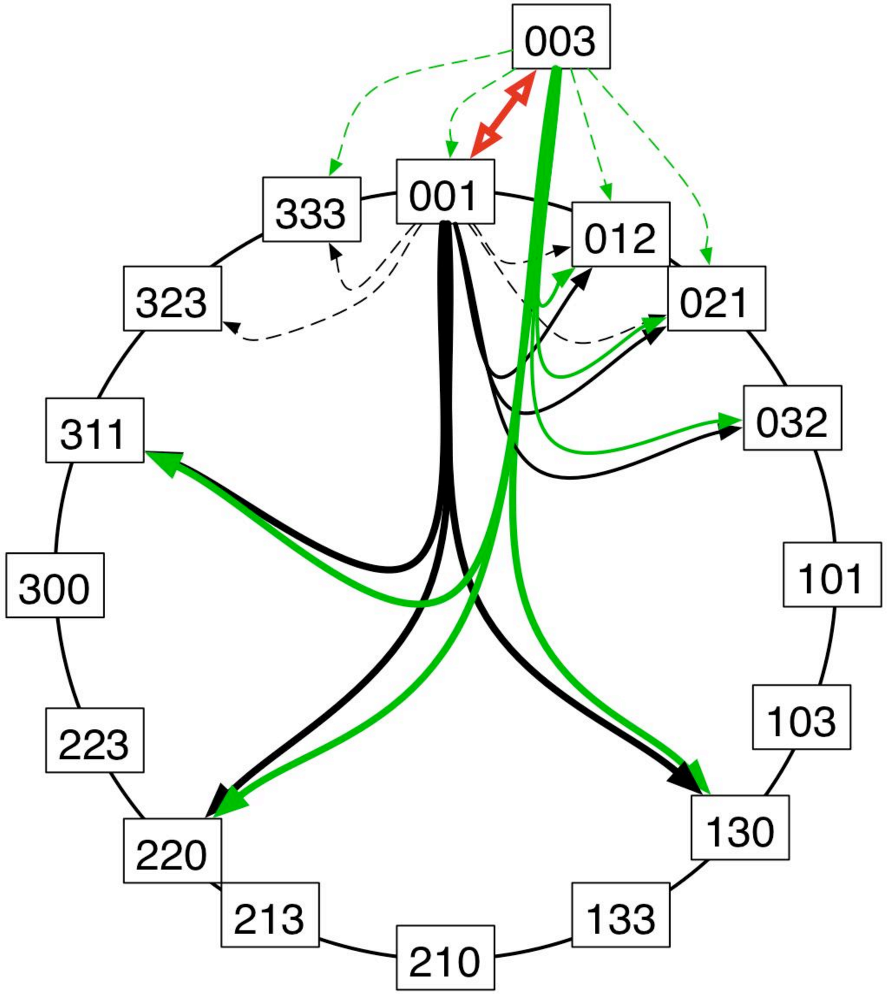
$$e^{-n/2^{bm}} \leq e^{-n/2^{c \log n}}$$
$$\leq e^{-n/n^c} \leq e^{-n^{c-1}}$$

- With (extremely) high probability there is no peer with the same prefix of length $(1+\varepsilon)(\log n)/b$
- Hence we have $(1+\varepsilon)(\log n)/b$ rows with $2^b$-1 entries each

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | 1x | 2x | 3x | 4x | 5x | | 7x | 8x | 9x | ax | bx | cx | dx | ex | fx |
| 60x | 61x | 62x | 63x | 64x | | 66x | 67x | 68x | 69x | 6ax | 6bx | 6cx | 6dx | 6ex | 6fx |
| 650x | 651x | 652x | 653x | 654x | 655x | 656x | 657x | 658x | 659x | | 65bx | 65cx | 65dx | 65ex | 65fx |
| 65a0x | | 65a2x | 65a3x | 65a4x | 65a5x | 65a6x | 65a7x | 65a8x | 65a9x | 65aax | 65abx | 65acx | 65adx | 65aex | 65afx |

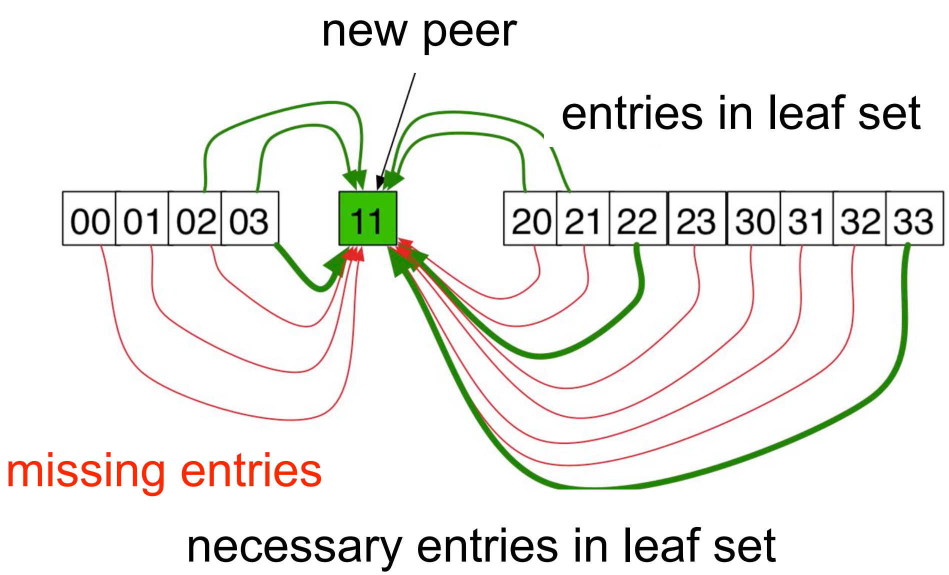
# A Peer Enters

- **New node x sends message to the node z with the longest common prefix p**
- **x receives**
  - routing table of z
  - leaf set of z
- **z updates leaf-set**
- **x informs $\ell$-leaf set**
- **x informs peers in routing table**
  - with same prefix p (if $\ell/2 < 2^b$)
- **Numbor of messages for adding a peer**
  - $\ell$ messages to the leaf-set
  - expected ($2^b$ - $\ell$/2) messages to nodes with common prefix
  - one message to z with answer



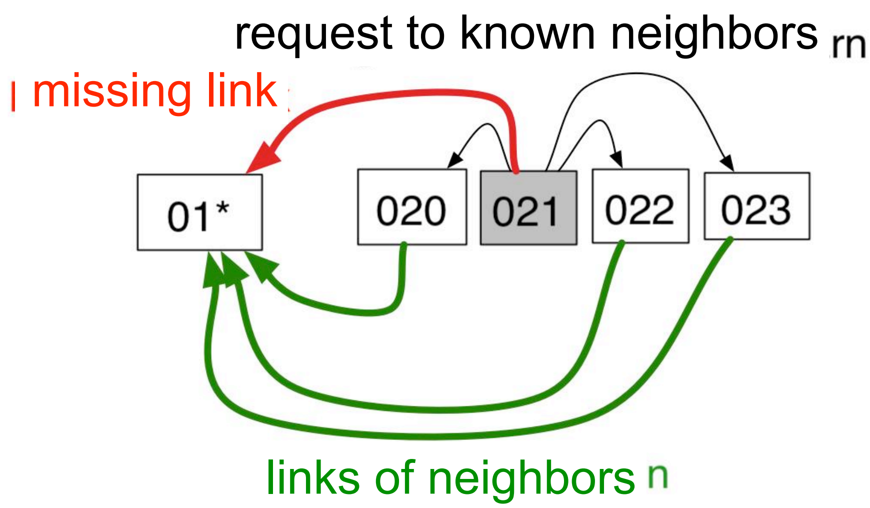
# When the Entry-Operation Errs

- **Inheriting the next neighbor routing table does not allows work perfectly**
- **Example**
  - If no peer with 1* exists then all other peers have to point to the new node
  - Inserting 11
  - 03 knows from its routing table
    - 22,33
    - 00,01,02
  - 02 knows from the leaf-set
    - 01,02,20,21
- **11 cannot add all necessary links to the routing tables**



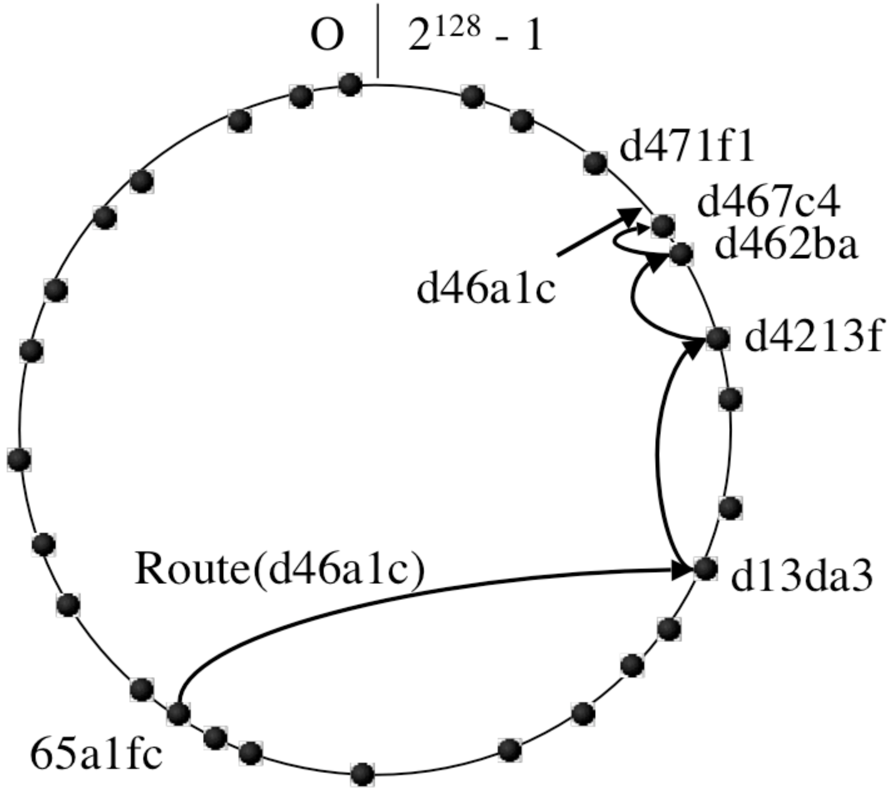
# Missing Entries in the Routing Table

- **Assume the entry $R_i^j$ is missing at peer D**
  - j-th row and i-th column of the routing table
- **This is noticed if a message of a peer with such a prefix is received**
- **This may also happen if a peer leaves the network**
- **Contact peers in the same row**
  - if they know a peer this address is copied
- **If this fails then perform routing to the missing link**



# Lookup

- **Compute the target ID using the hash function**
- **If the address is within the $\ell$-leaf set**
  - the message is sent directly
  - or it discovers that the target is missing
- **Else use the address in the routing table to forward the mesage**
- **If this fails take best fit from all addresses**



# Lookup in Detail

- **L:** **$\ell$-leafset**
- **R:** **routing table**
- **M:** **nodes in the vicinity of D (according to RTT)**
- **D:** **key**
- **A:** **nodeID of current peer**
- **$R^i_l$:** **j-th row and i-th column of the routing table**
- **$L_i$:** **numbering of the leaf set**
- **$D_i$:** **i-th digit of key D**
- **shl(A):** **length of the largest common prefix of A and D (shared header length)**

if $(L_{-\lfloor |L|/2 \rfloor} \leq D \leq L_{\lfloor |L|/2 \rfloor})$ {
    // $D$ is within range of our leaf set
    forward to $L_i$, s.th. $|D - L_i|$ is minimal;
} else {
    // use the routing table
    Let $l = shl(D, A)$;
    if $(R_l^{D_l} \neq null)$ {
        forward to $R_l^{D_l}$;
    }
    else {
        // rare case
        forward to $T \in L \cup R \cup M$, s.th.
            $shl(T, D) \geq l$,
            $|T - D| < |A - D|$
    }
}

# Routing — Discussion

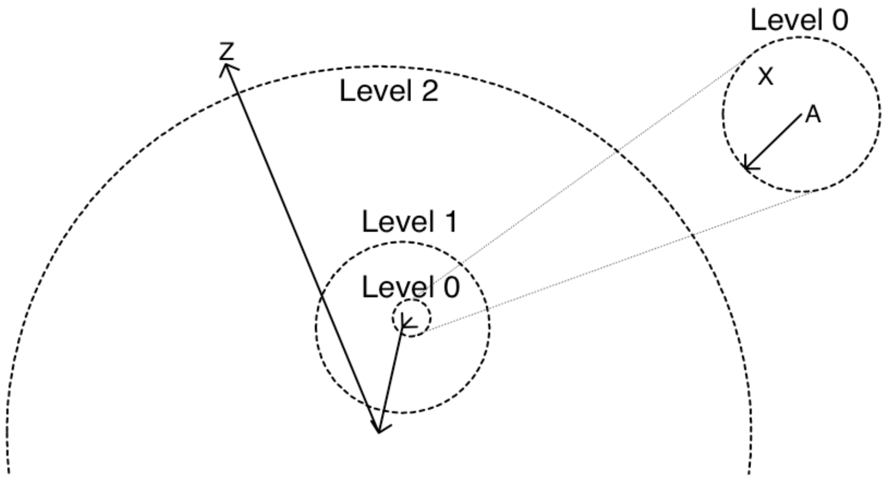
- **If the Routing-Table is correct**
  - routing needs O((log n)/b) messages
- **As long as the leaf-set is correct**
  - routing needs O(n/l) messages
  - unrealistic worst case since even damaged routing tables allow dramatic speedup
- **Routing does not use the real distances**
  - M is used only if errors in the routing table occur
  - using locality improvements are possible
- **Thus, Pastry uses heuristics for improving the lookup time**
  - these are applied to the last, most expensive, hops

# Localization of the k Nearest Peers

- **Leaf-set peers are not near, e.g.**
  - New Zealand, California, India, ...
- **TCP protocol measures latency**
  - latencies (RTT) can define a metric
  - this forms the foundation for finding the nearest peers
- **All methods of Pastry are based on heuristics**
  - i.e. no rigorous (mathematical) proof of efficiency
- **Assumption: metric is Euclidean**

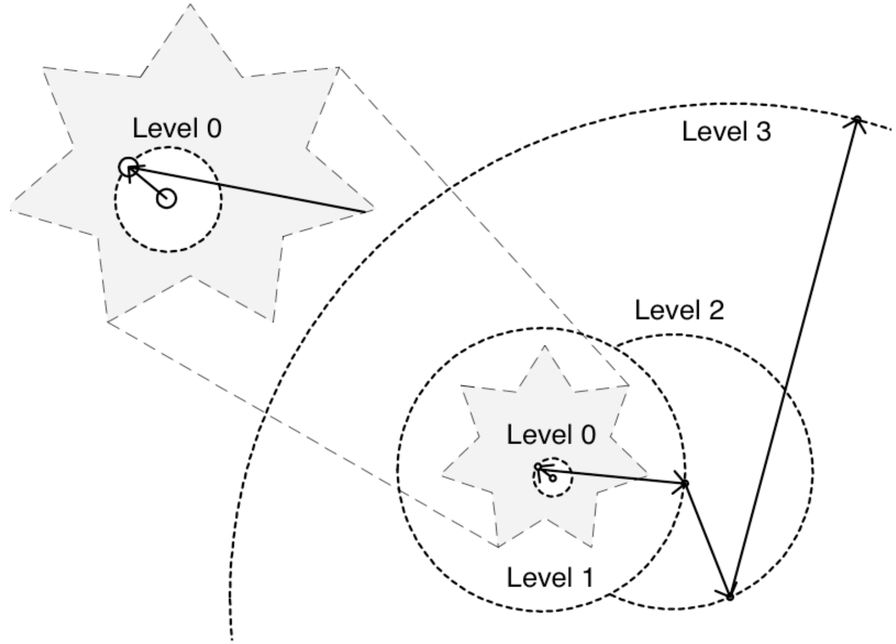
# Locality in the Routing Table

- **Assumption**
  - When a peer is inserted the peers contacts a near peer
  - All peers have optimized routing tables
- **But:**
  - The first contact is not necessary near according to the node-ID
- **1st step**
  - Copy entries of the first row of the routing table of P
    - good approximation because of the triangle inequality (metric)
- **2nd step**
  - Contact fitting peer p‘ of p with the same first letter
  - Again the entries are relatively close
- **Repeat these steps until all entries are updated**



# Locality in the Routing Table

- **In the best case**
  - each entry in the routing table is optimal w.r.t. distance metric
  - this does not lead to the shortest path
- **There is hope for short lookup times**
  - with the length of the common prefix the latency metric grows exponentially
  - the last hops are the most expensive ones
  - here the leaf-set entries help

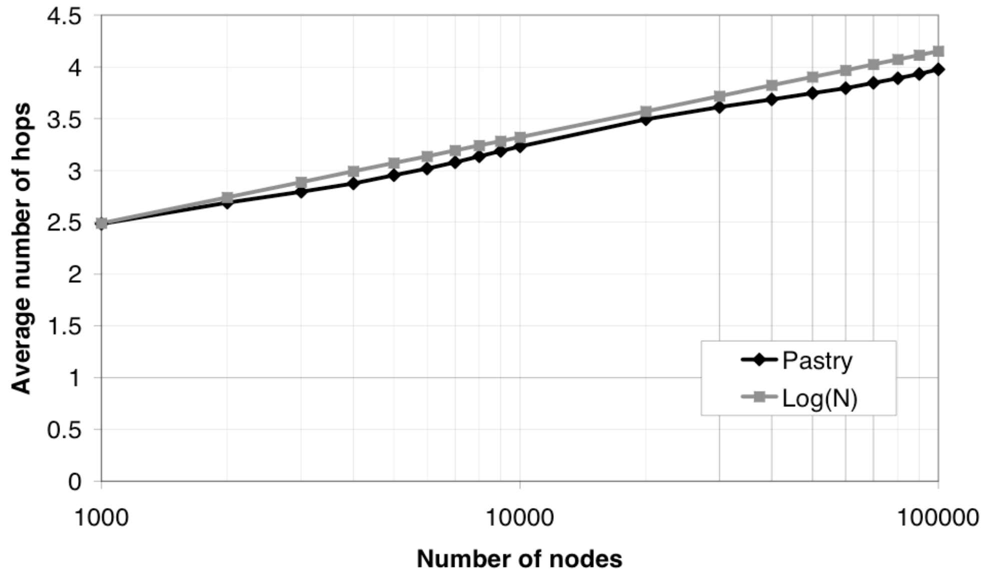


# Localization of Near Nodes

- **Node-ID metric and latency metric are not compatible**
- **If data is replicated on k peers then peers with similar Node-ID might be missed**
- **Here, a heuristic is used**
- **Experiments validate this approach**

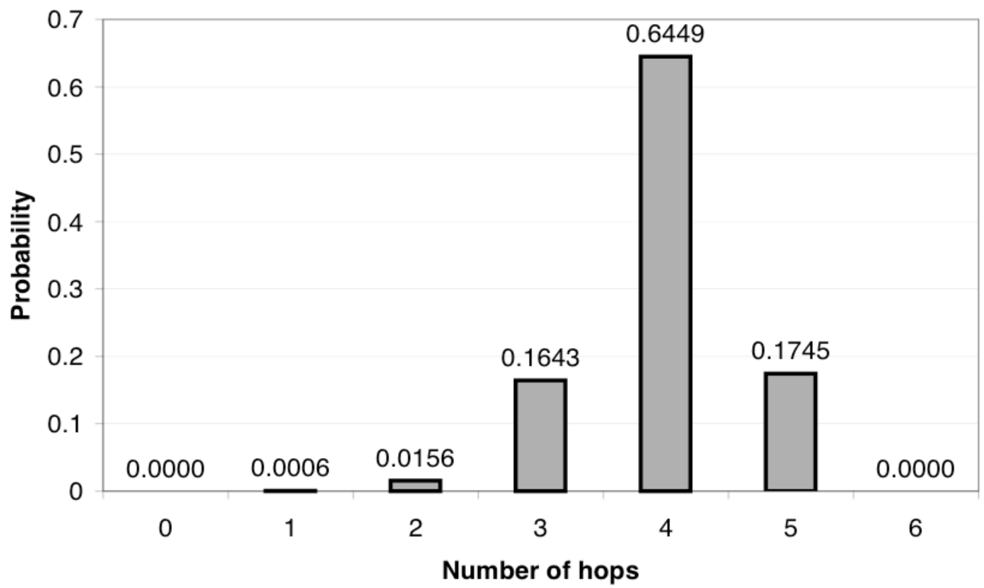
# Experimental Results — Scalability

- **Parameter b=4, l=16, M=32**
- **In this experiment the hop distance grows logarithmically with the number of nodes**
- **The analysis predicts O(log n)**
- **Fits well**



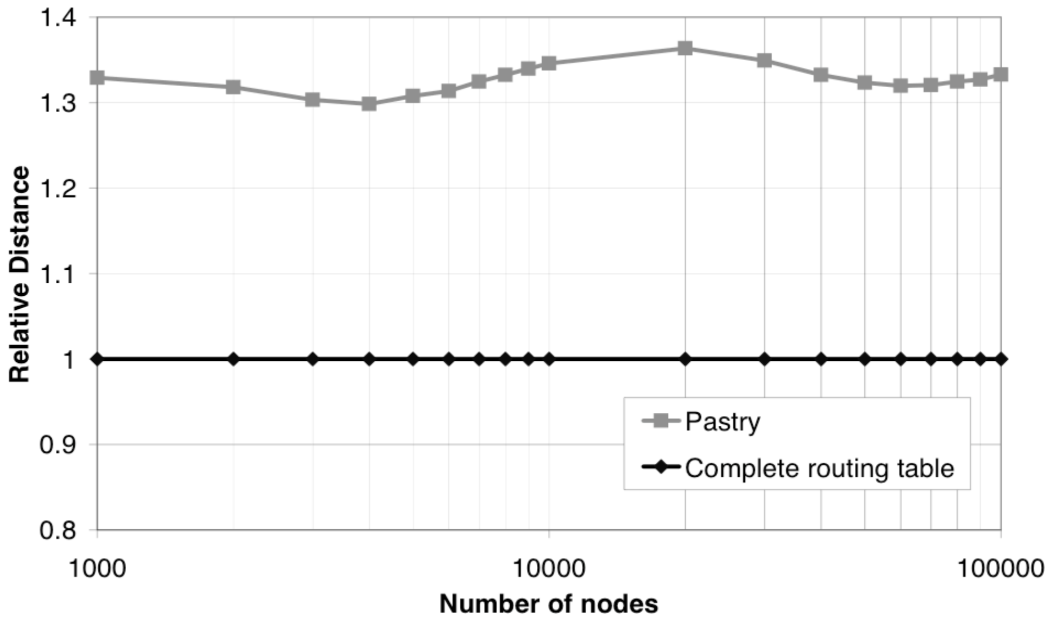
# Experimental Results
# Distribution of Hops

- **Parameter b=4, l=16, M=32, n = 100,000**
- **Result**
  - deviation from the expected hop distance is extremely small
- **Analysis predicts difference with extremely small probability**
  - fits well



# Experimental Results — Latency

- **Parameter b=4, l=16, M=3**
- **Compared to the shortest path astonishingly small**
  - seems to be constant

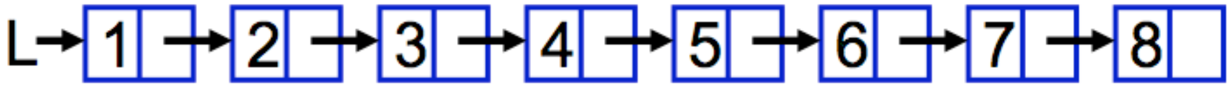


# Skip-Net

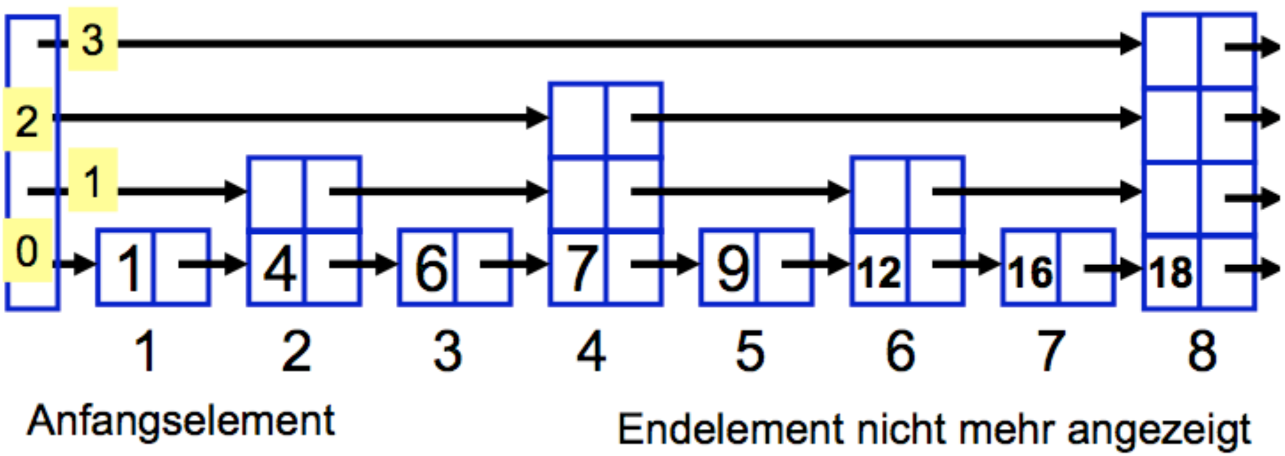
- **J. Aspnes and G. Shah. Skip graphs, 2003**
- **SkipNet: A Scalable Overlay Network with Practical Locality Properties Nicholas J.A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, Alec Wolman, 2003**
- **Problem:**
  - Ordered storage of data on peers
  - without complicated balancing
- **Solution**
  - Skip-graphs

# Skip-Lists

Einfach verkettete Liste:



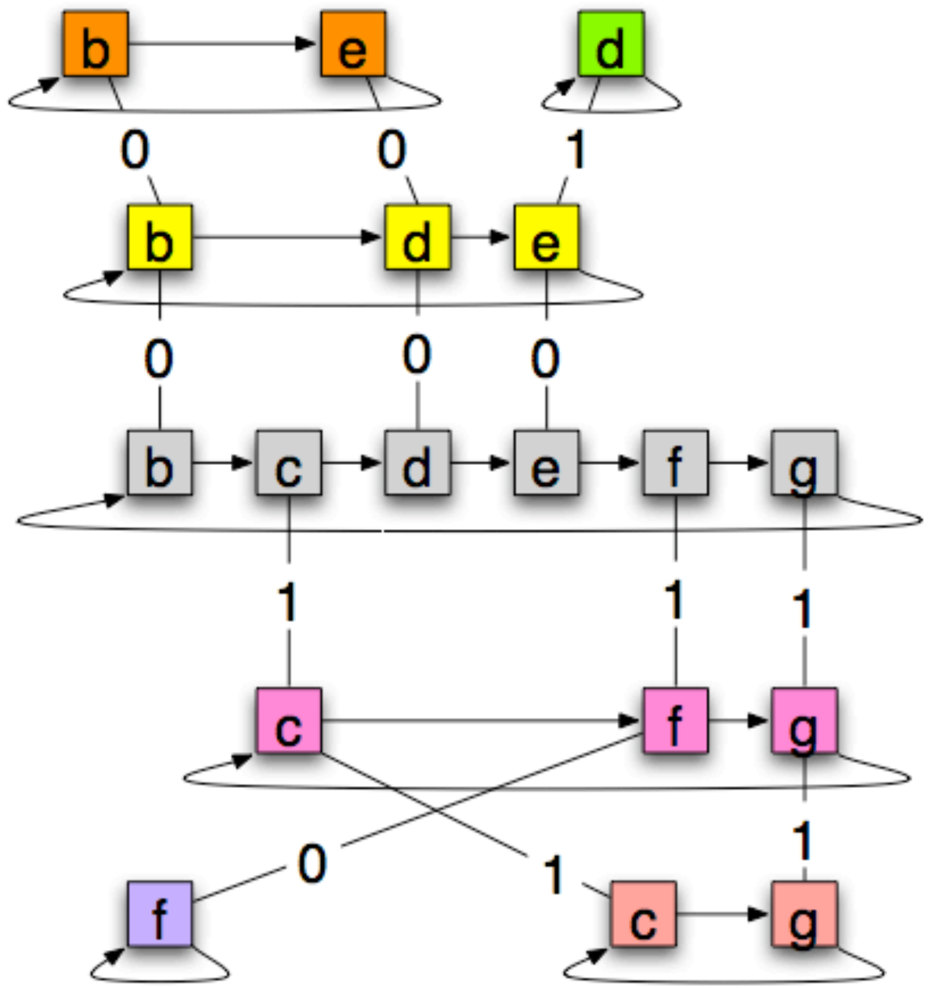

Niveau i des Zeigers



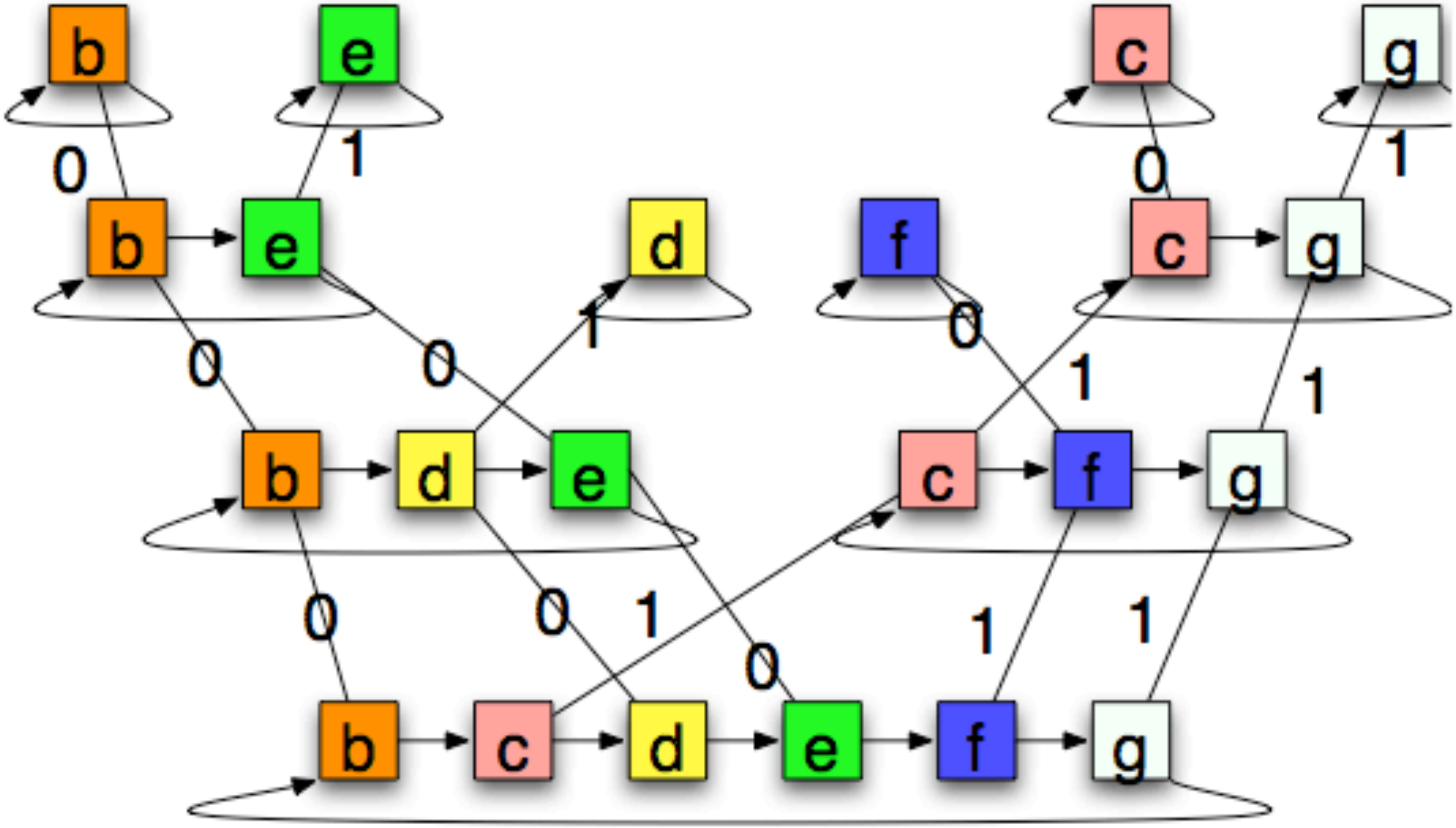

Bilder aus Petra Mutzels Vorlesung, Unversität Dortmund

# Skip-Graphs

- **J. Aspnes and G. Shah. Skip graphs, 2003**
- **Idea**
  - „Heads“ and „Tails“ of a coin toss recursively participate in an own game
- **Properties**
  - higly resilient
  - Diameter and degree O(log n) with high probability
  - Ordering of data remains

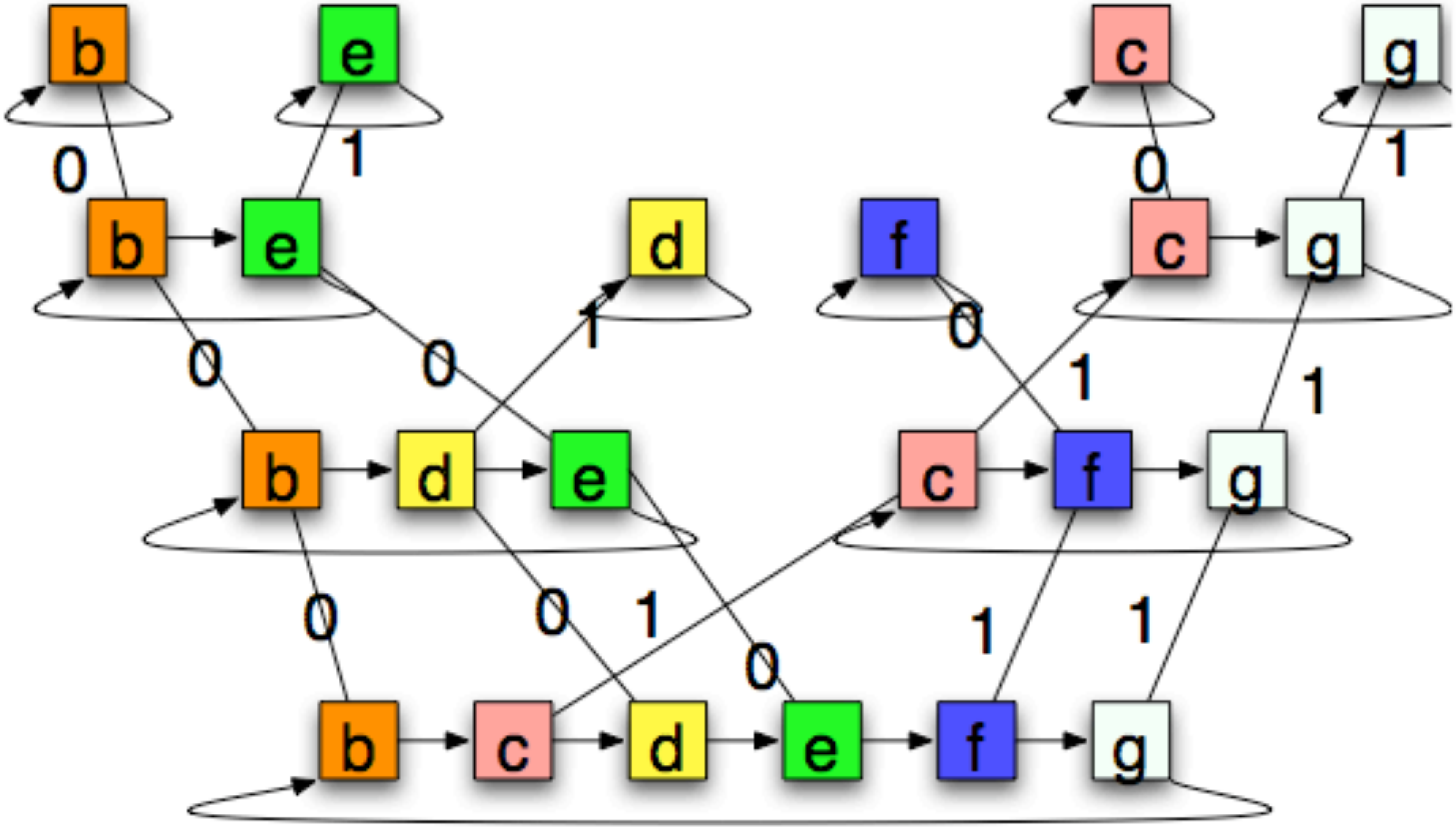


# Search for Name-ID



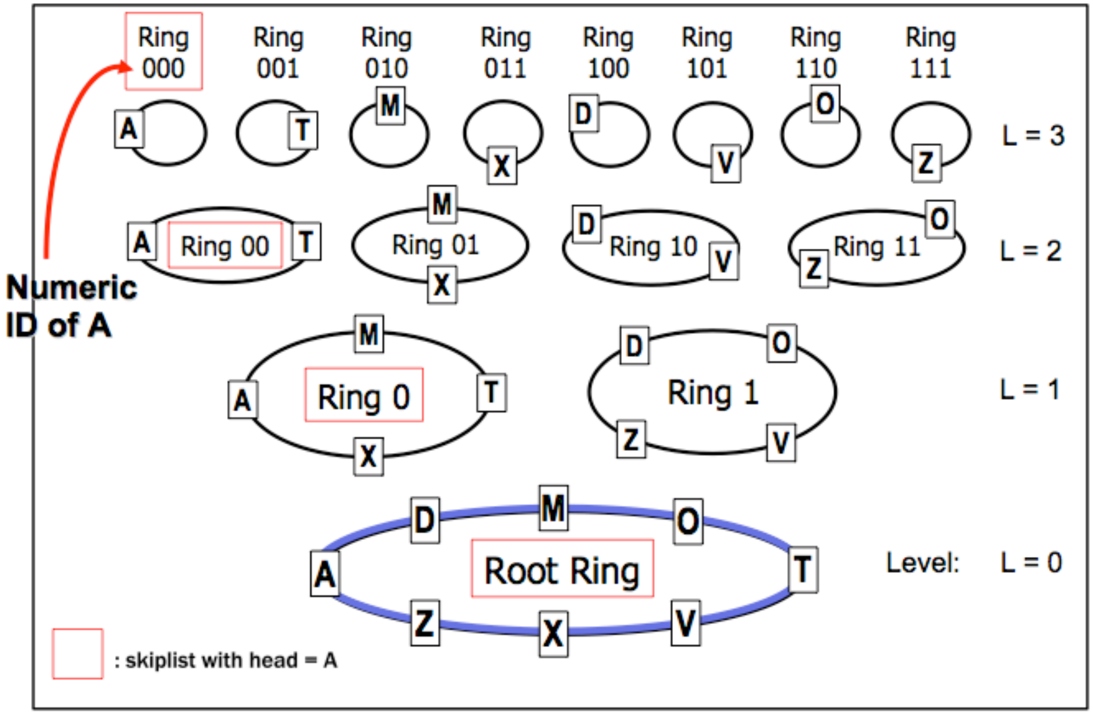
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

# Search for Num-ID



# Alternative Representation

- **From: P2P Network Structured Networks von Pedro Garcia Lopez, Universitat Rovira I Virgili**



# Inserting Peers

- **J. Aspnes and G. Shah. Skip graphs, 2003**

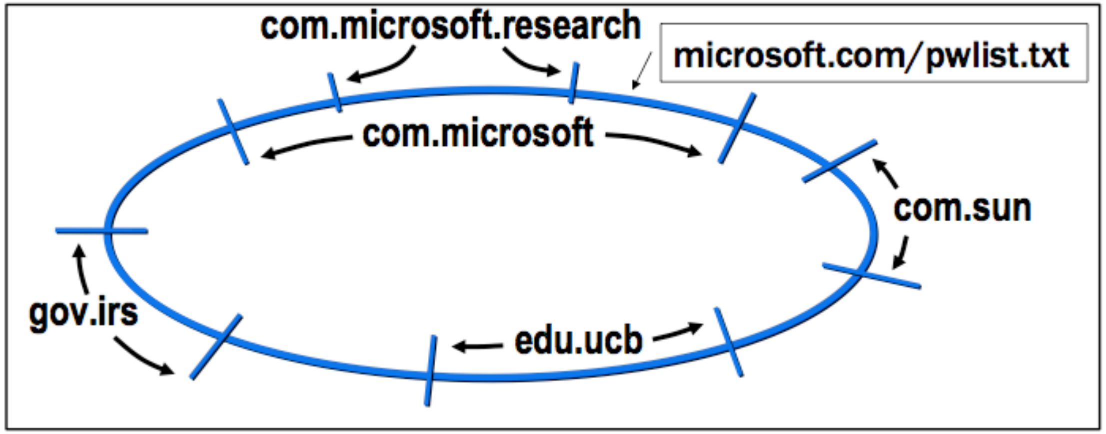
- **Algorithm**
  - Lookup of correct place according to node name
  - Insertion into higher ranks

- **Runtime: O(log n) hops and O(log n) messages with high probability**

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

# Locality of Content and Routing

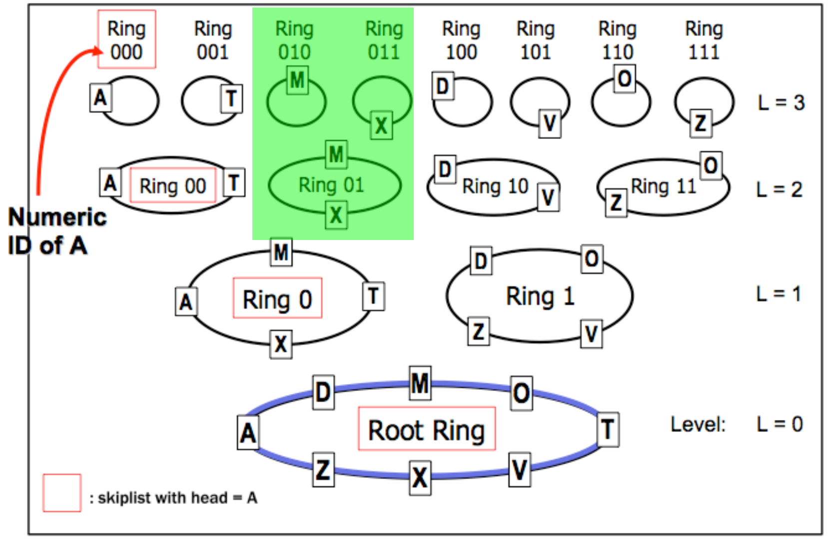
- **Locality of content**
  - underlying ordering
- **Alternative mapping of data**
  - data can be stored using num-id
- **Locality of Routing**
  - if the hosts are sorting along domains then local routing within a domain can be facilitated where possible

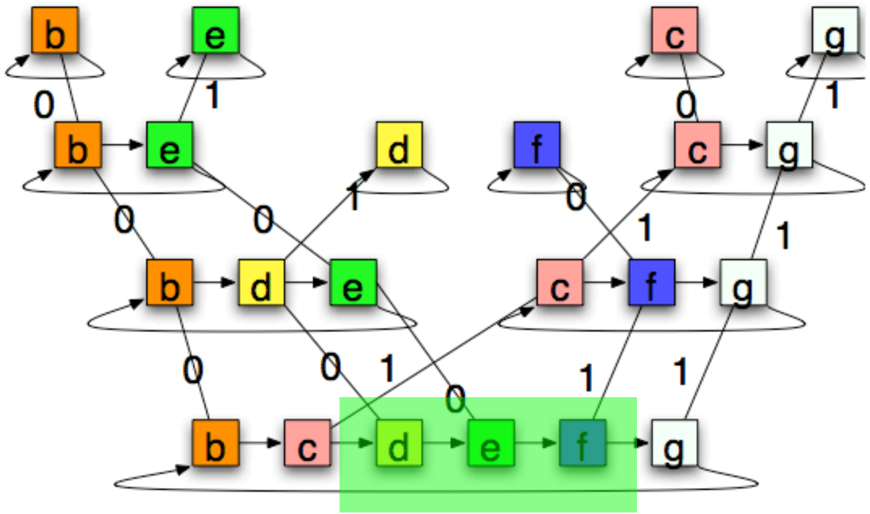


# Range Search

- **Num-ID range search**
- **Name-ID range search**
- **Intersection of Num-ID and Name-ID**

- **Running time:**
  - O(log n) for first element
  - Then constant time for each successing elements



# New Trends for Locality of Peer-to-Peer Networks

- **RTT gives a distance measure between nodes of the Internet**
- **More than 5% of all triples of nodes in the Internet violate the triangle inequality (TIV)**
- **More than 50% of all pairs of nodes form an edge of a TIV**
  - Wang, G., Zhang, B., Ng, T.S.E.: Towards network triangle in equality violation aware distributed systems. In:IMC.(2007)
- **Better paths are possible using Peer-to-Peer Networks**
  - Lumezanu, C., Levin, D., Spring, N.: PeerWise discovery and negotiation of faster paths. In: HotNets. (2007)

# Reasons for TIV

Triangle Inequality and Routing Policy Violations in the Internet
Cristian Lumezanu, Randy Baden, Neil Spring, and Bobby Bhattacharjee, 2009

| | | |
|---|---|---|
| **Total Detours** | | 793,693 |
| **Impossible AS Paths** | | 460,830 (58%) |
| Cause | Customer transit | 343,381 (75%) |
| | Peer transit | 117,449 (25%) |
| Type | Truly disjoint | 302,207 (66%) |
| | Borderline | 153,057 (33%) |
| | Undercover | 5,503 (1%) |
| **Possible AS Paths** | | 197,453 (25%) |
| Traffic Eng. | Relay AS not on direct path | 56,813 (29%) |
| | Direct, detour paths differ | 103,215 (52%) |
| | Direct, detour paths same | 37,425 (19%) |
| Path length | Shorter than direct | 17,770 (9%) |
| | Equal to direct | 75,032 (38%) |
| | Longer than direct | 104,651 (53%) |
| Transit cost | Smaller than direct | 35,541 (18%) |
| | Equal to direct | 96,751 (49%) |
| | Greater than direct | 65,161 (33%) |
| **Unknown** | | 135,410 (17%) |

**Table 1.** Detour paths are *possible* (may be available to the BGP decision process) or *impossible* (not advertised by BGP). Percentages inside the tables are relative to the total possible or impossible paths. Categories separated by horizontal lines overlap.

# P2P can be faster than IP

- Lumezanu, C., Levin, D., Spring, N.: PeerWise discovery and negotiation of faster paths. In: HotNets. (2007)