DAAD Summerschool Curitiba 2011

Aspects of Large Scale High Speed Computing Building Blocks of a Cloud

Storage Networks

5: Peer-to-Peer Networks

Christian Schindelhauer

Technical Faculty

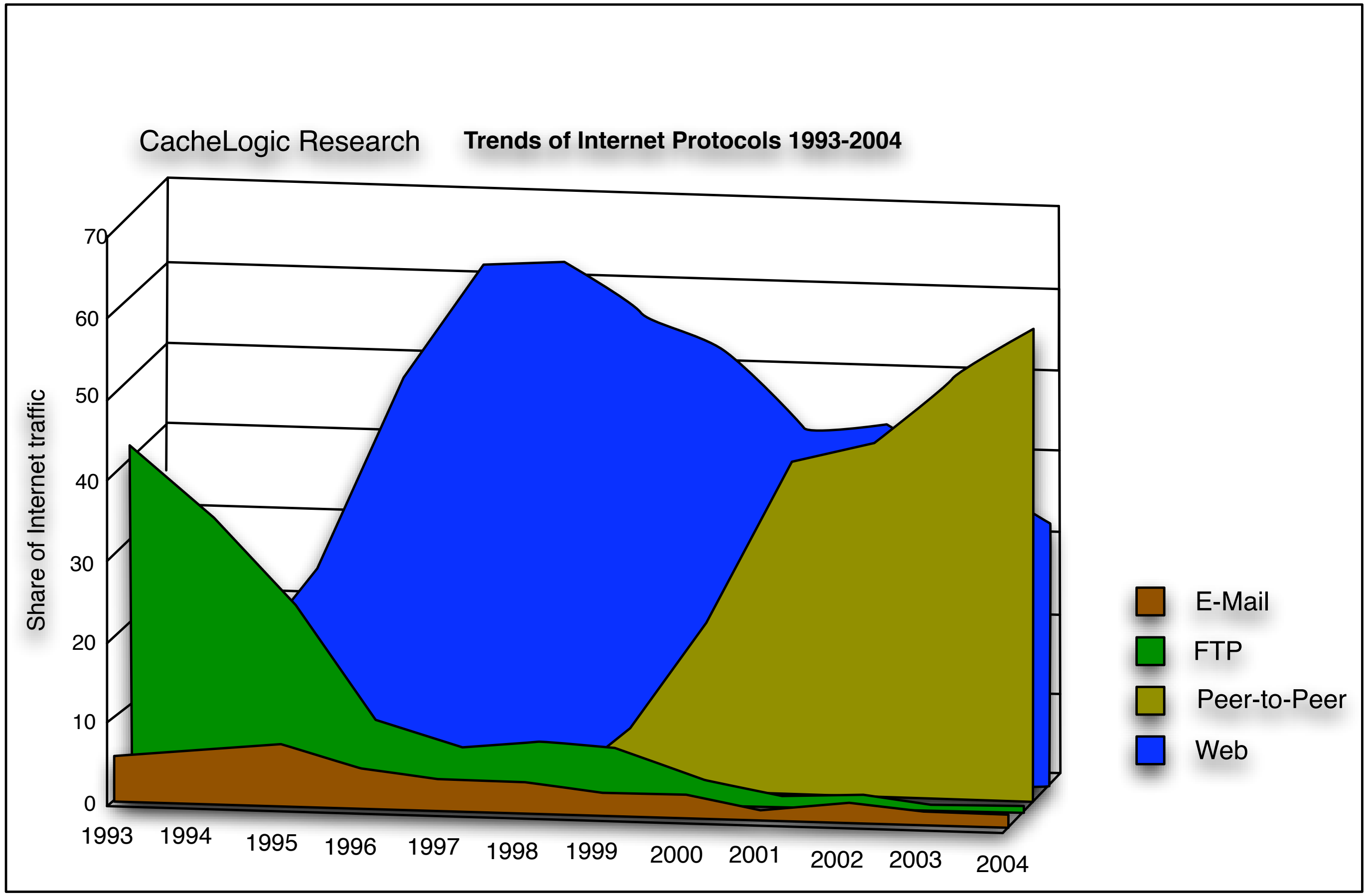
Computer-Networks and Telematics

University of Freiburg

Outline

- Principles and history
- Algorithms and Methods
  - DHTs
  - Chord
  - Pastry

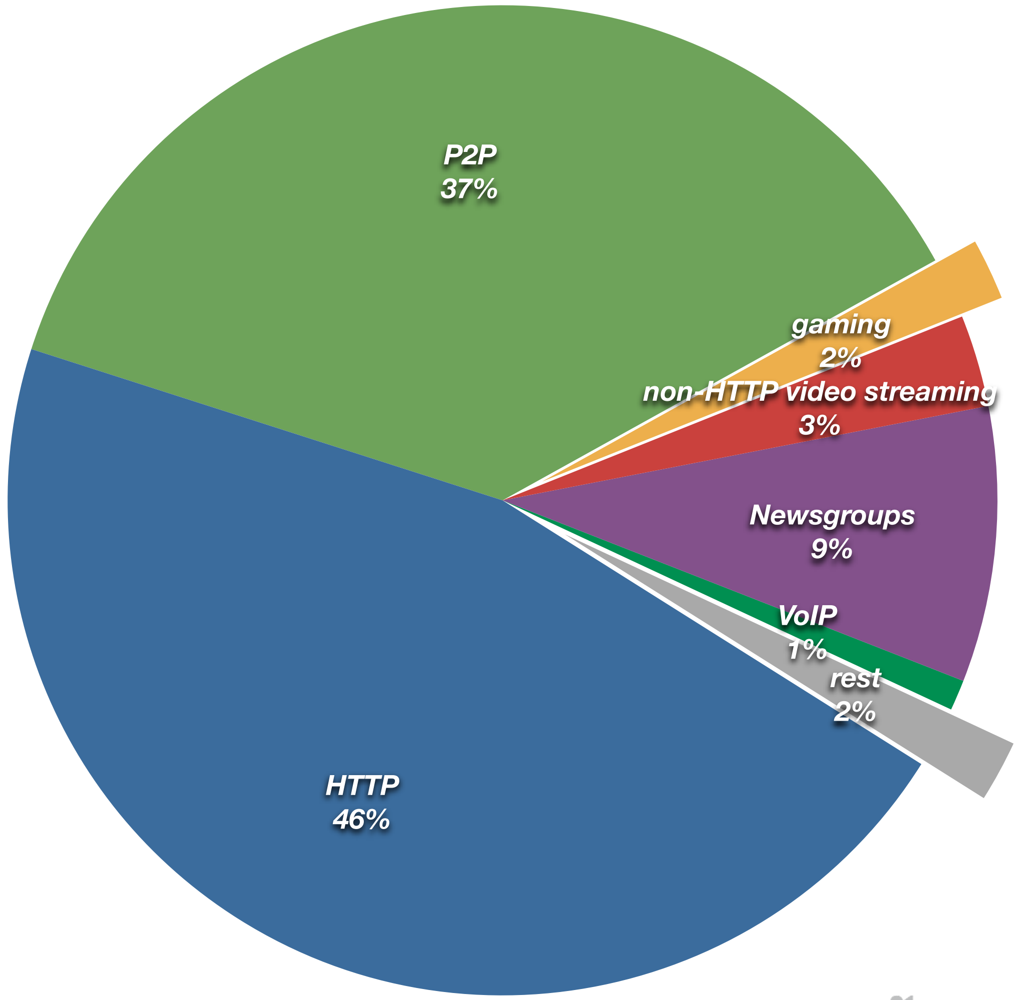
Global Internet Traffic Shares 1993-2004



Source: CacheLogic 2005

Global Internet Traffic 2007

- **Ellacoya report (June 2007)**
  - worldwide HTTP traffic volume overtakes P2P after four years continues record
- **Main reason: Youtube.com**



Milestones P2P Systems

- Napster (1st version: 1999-2000)
- Gnutella (2000), Gnutella-2 (2002)
- Edonkey (2000)
  - later: Overnet usese Kademlia
- FreeNet (2000)
  - Anonymized download
- JXTA (2001)
  - Open source P2P network platform
- FastTrack (2001)
  - known from KaZaa, Morpheus, Grokster
- Bittorrent (2001)
  - only download, no search
- Skype (2003)
  - VoIP (voice over IP), Chat, Video

Milestones Theory

- Distributed Hash-Tables (DHT) (1997)
  - introduced for load balancing between web-servers
- CAN (2001)
  - efficient distributed DHT data structure for P2P networks
- Chord (2001)
  - efficient distributed P2P network with logarithmic search time
- Pastry/Tapestry (2001)
  - efficient distributed P2P network using Plaxton routing
- Kademlia (2002)
  - P2P-Lookup based on XOr-Metrik
- Many more exciting approaches
  - Viceroy, Distance-Halving, Koorde, Skip-Net, P-Grid, ...
- Recent developments
  - Network Coding for P2P
  - Game theory in P2P
  - Anonymity, Security

CoNe Freiburg

What is a P2P Network?

- What is P2P NOT?
  - a peer-to-peer network is **not a client-server network**
- Etymology: peer
  - from latin par = equal
  - one that is of equal standing with another
  - P2P, Peer-to-Peer: a relationship between equal partners
- Definition
  - a Peer-to-Peer Network is a communication network between computers in the Internet
    - without central control
    - and without reliable partners
- Observation
  - the Internet can be seen as a large P2P network

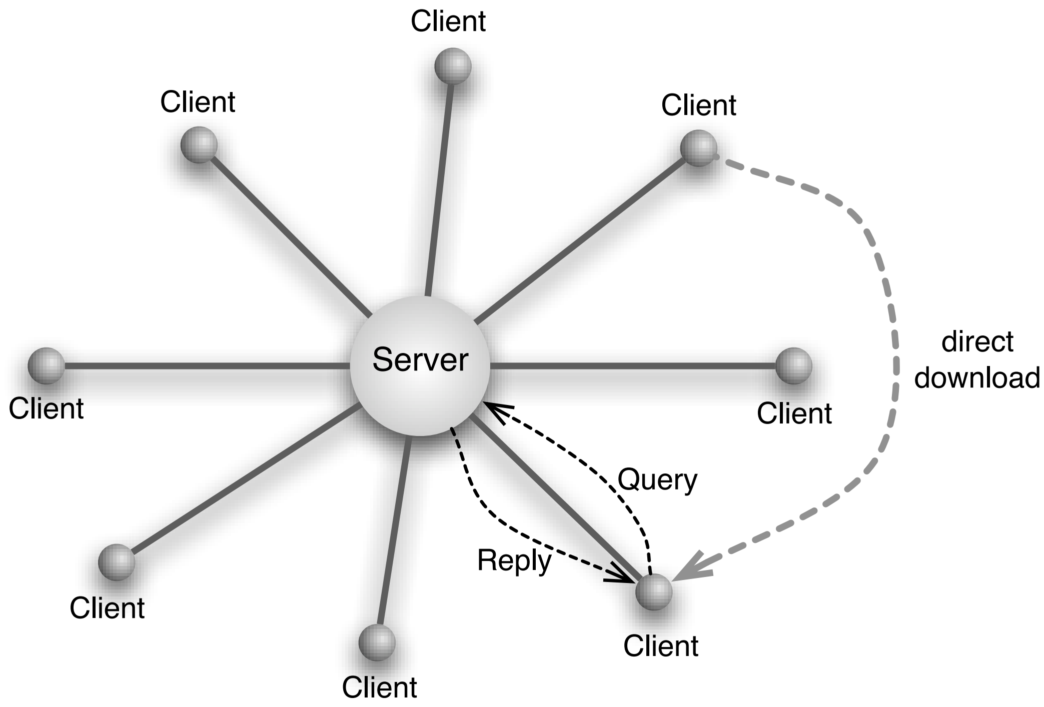
UNI FREIBURG

Napster

- Shawn (Napster) Fanning
  - published 1999 his beta version of the now legendary Napster P2P network
  - File-sharing-System
  - Used as mp3 distribution system
  - In autumn 1999 Napster has been called download of the year
- Copyright infringement lawsuit of the music industry in June 2000
- End of 2000: cooperation deal
  - between Fanning and Bertelsmann Ecommerce
- Since then Napster is a commercial file-sharing platform

How Did Napster Work?

- **Client-Server**
- **Server stores**
  - Index with meta-data
    - file name, date, etc
  - table of connections of participating clients
  - table of all files of participants
- **Query**
  - client queries file name
  - server looks up corresponding clients
  - server replies the owner of the file
  - querying client downloads the file from the file owning client

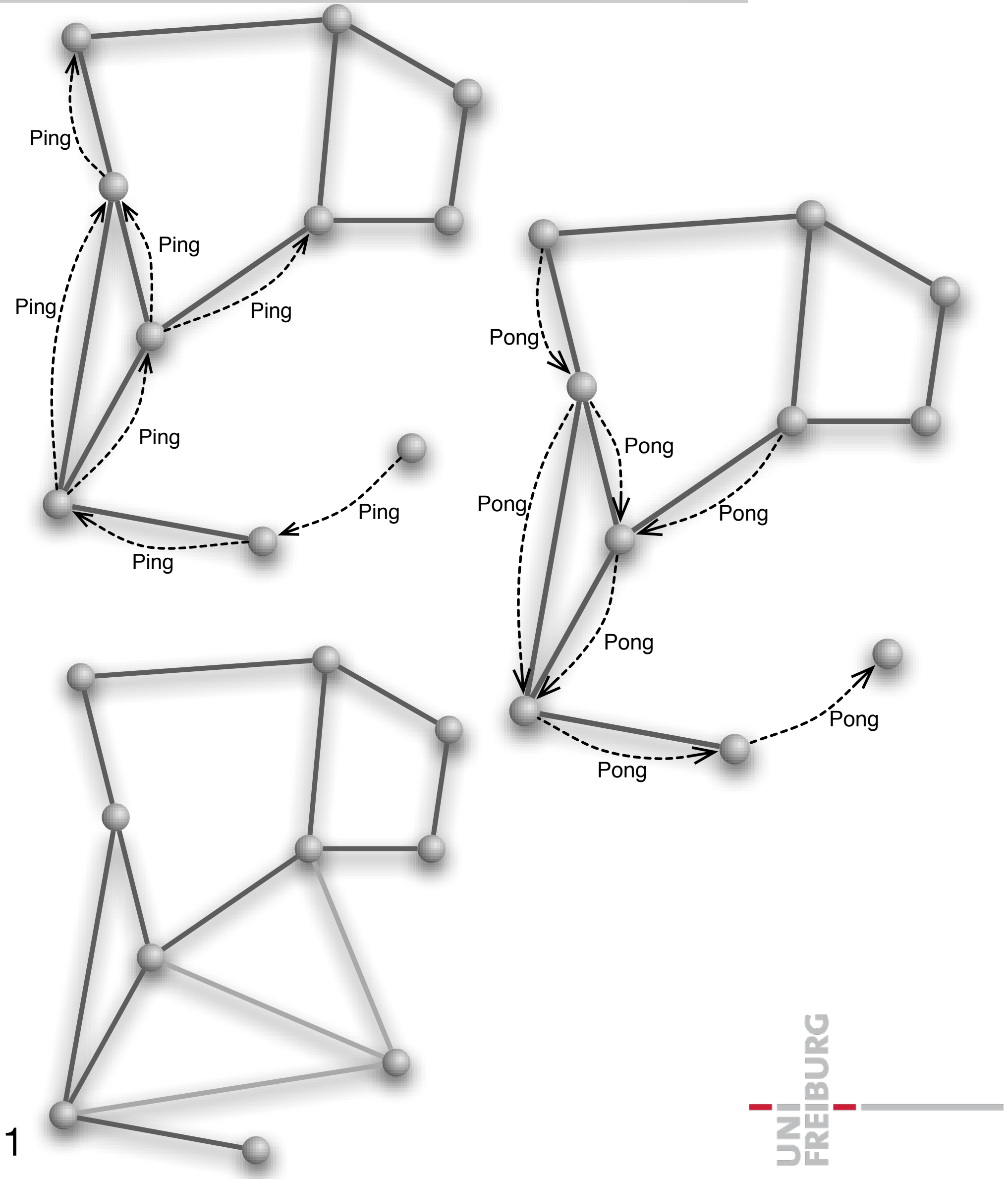


History of Gnutella

- Gnutella
  - was released in March 2000 by Justin Frankel and Tom Pepper from Nullsoft
  - Since 1999 Nullsoft is owned by AOL
- File-Sharing system
  - Same goal as Napster
  - But without any central structures

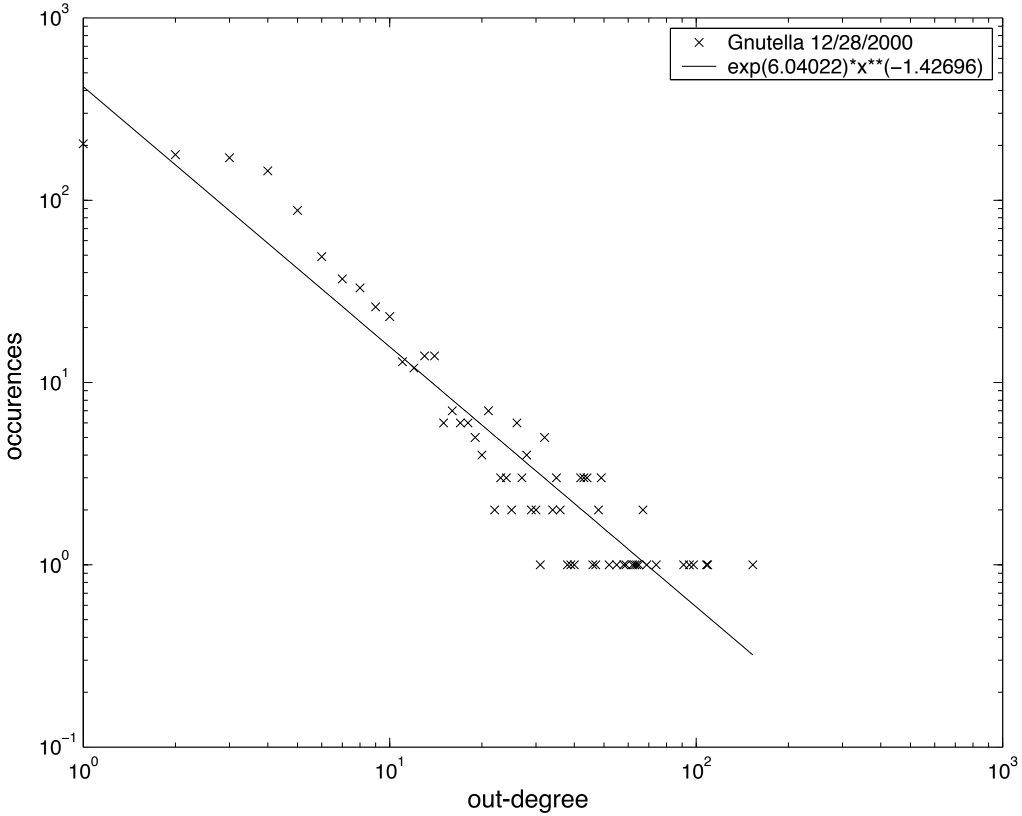
Gnutella — Connecting

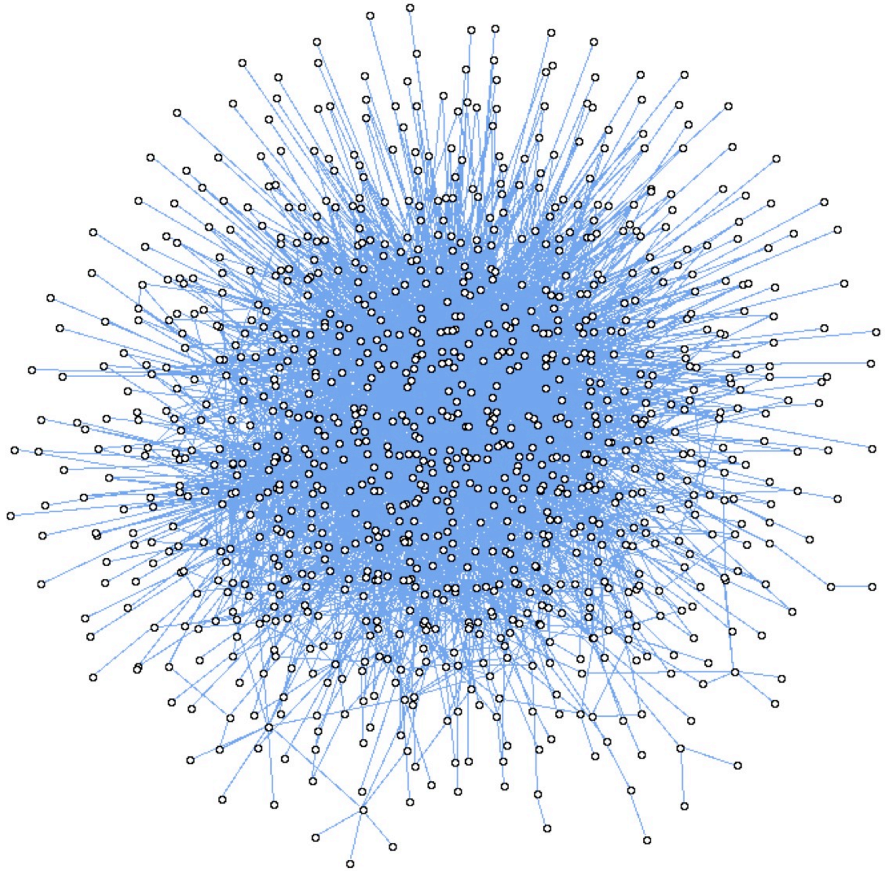
- **Neighbor lists**
  - Gnutella connects directly with other clients
  - the client software includes a list of usually online clients
  - the clients checks these clients until an active node has been found
  - an active client publishes its neighbor list
  - the query (ping) is forwarded to other nodes
  - the answer (pong) is sent back
  - neighbor lists are extended and stored
  - the number of the forwarding is limited (typically: five)



Gnutella — Graph Structure

- **Graph structure**
  - constructed by random process
  - underlies power law
  - without control

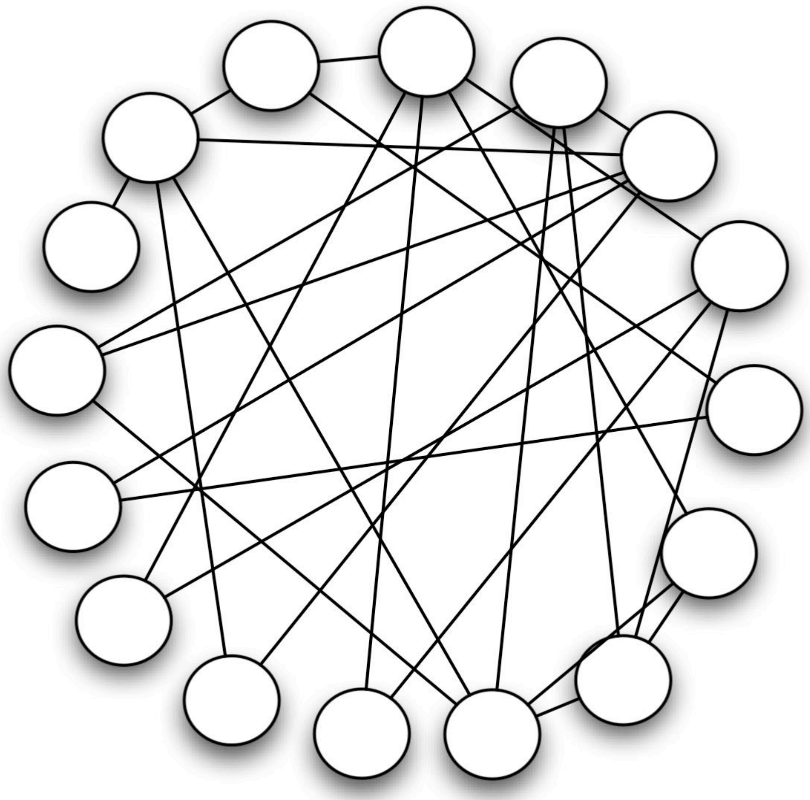




Gnutella snapshot in 2000

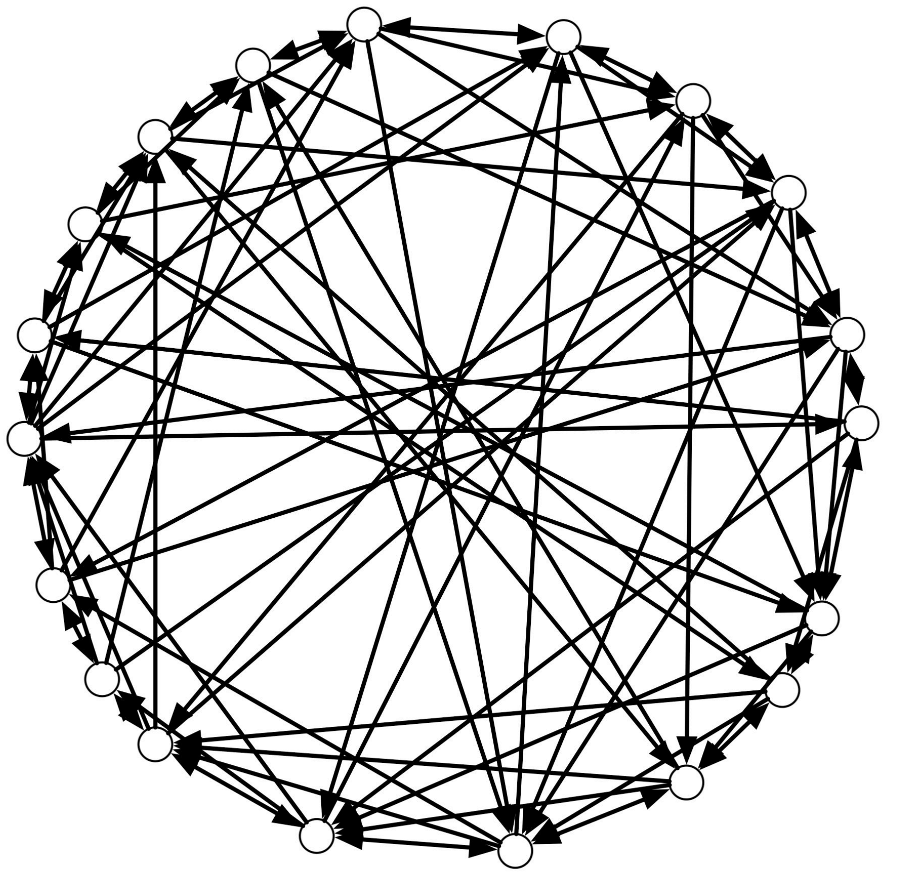
Why Gnutella Does Not Really Scale

- **Gnutella**
  - graph structure is random
  - degree of nodes is small
  - small diameter
  - strong connectivity
- **Lookup is expensive**
  - for finding an item the whole network must be searched
- **Gnutella‘s lookup does not scale**
  - reason: no structure within the index storage



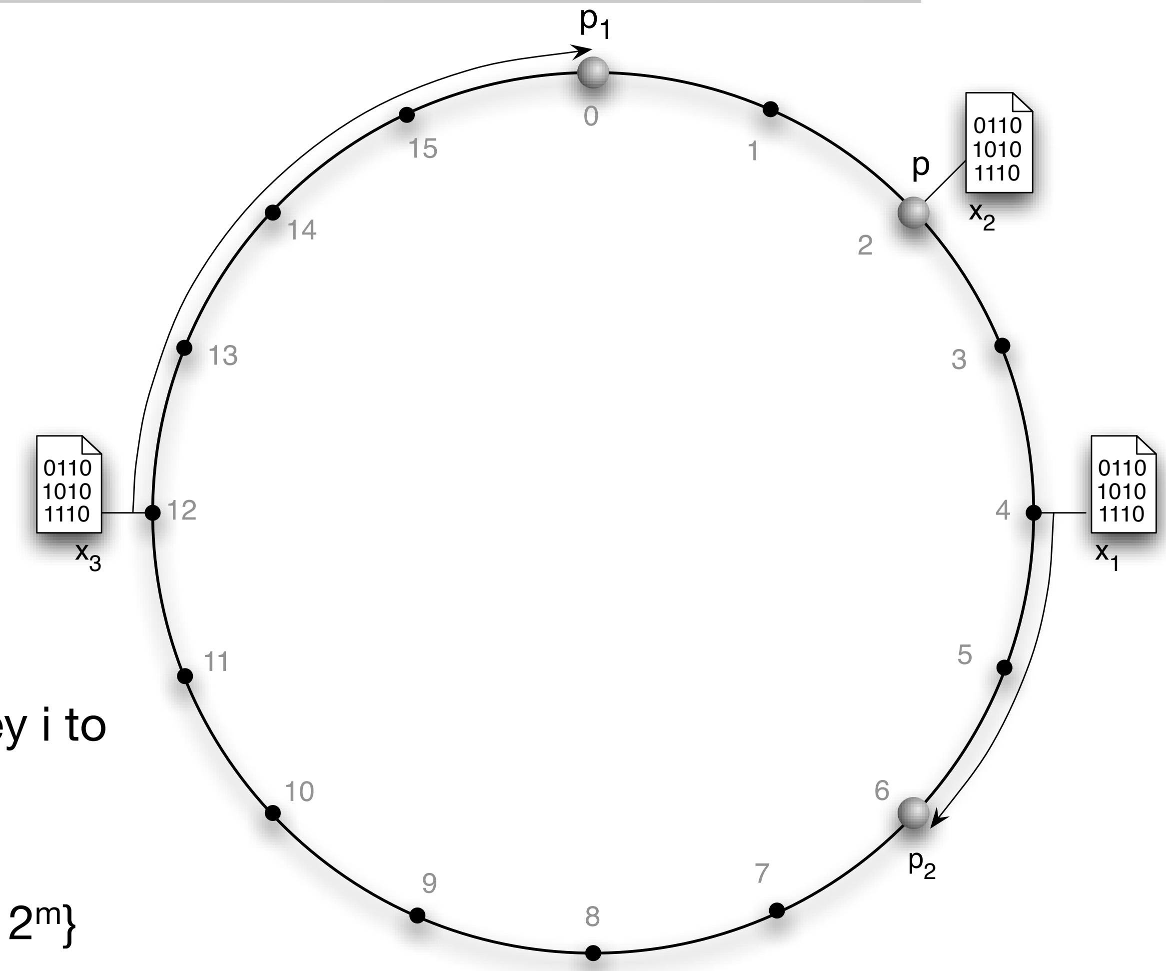
Chord

- **Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan (2001)**
- **Distributed Hash Table**
  - range {0,..,2^m-1}
  - for sufficient large m
- **Network**
  - ring-wise connections
  - shortcuts with exponential increasing distance



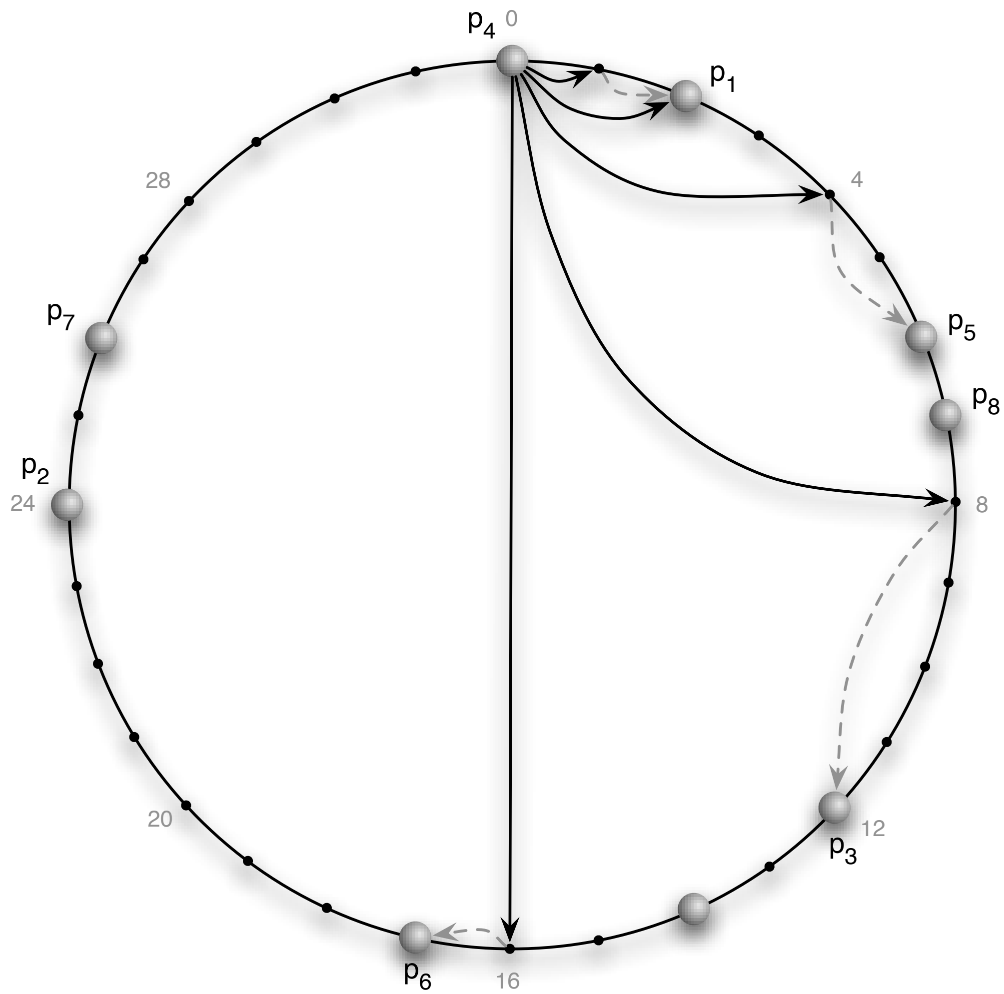
Chord as DHT

- **n number of peers**
- **V set of peers**
- **k number of data stored**
- **K set of stored data**
- **m: hash value length**
  - $m \geq 2 \log \max\{K,N\}$
- **Two hash functions mapping to $\{0,..,2^{m-1}\}$**
  - $r_V(b)$: maps peer to $\{0,..,2^{m-1}\}$
  - $r_K(i)$: maps index according to key i to $\{0,..,2^{m-1}\}$
- **Index i maps to peer $b = f_V(i)$**
  - $f_V(i) := \arg\min_{b \in V}\{(r_V(b)-r_K(i)) \bmod 2^m\}$



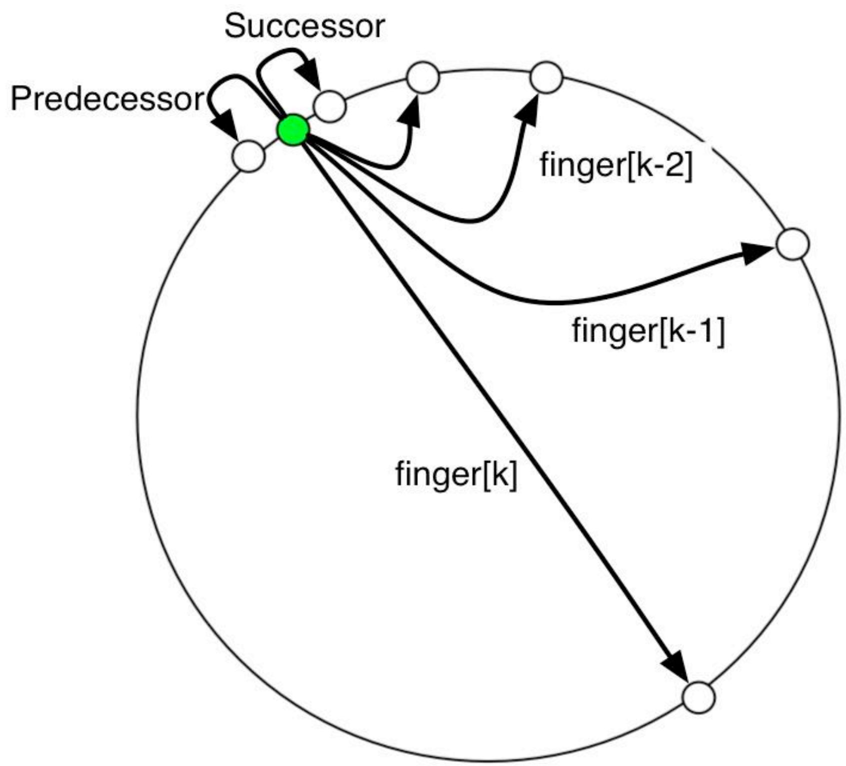
Pointer Structure of Chord

- **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. $1-n^{-c}$.



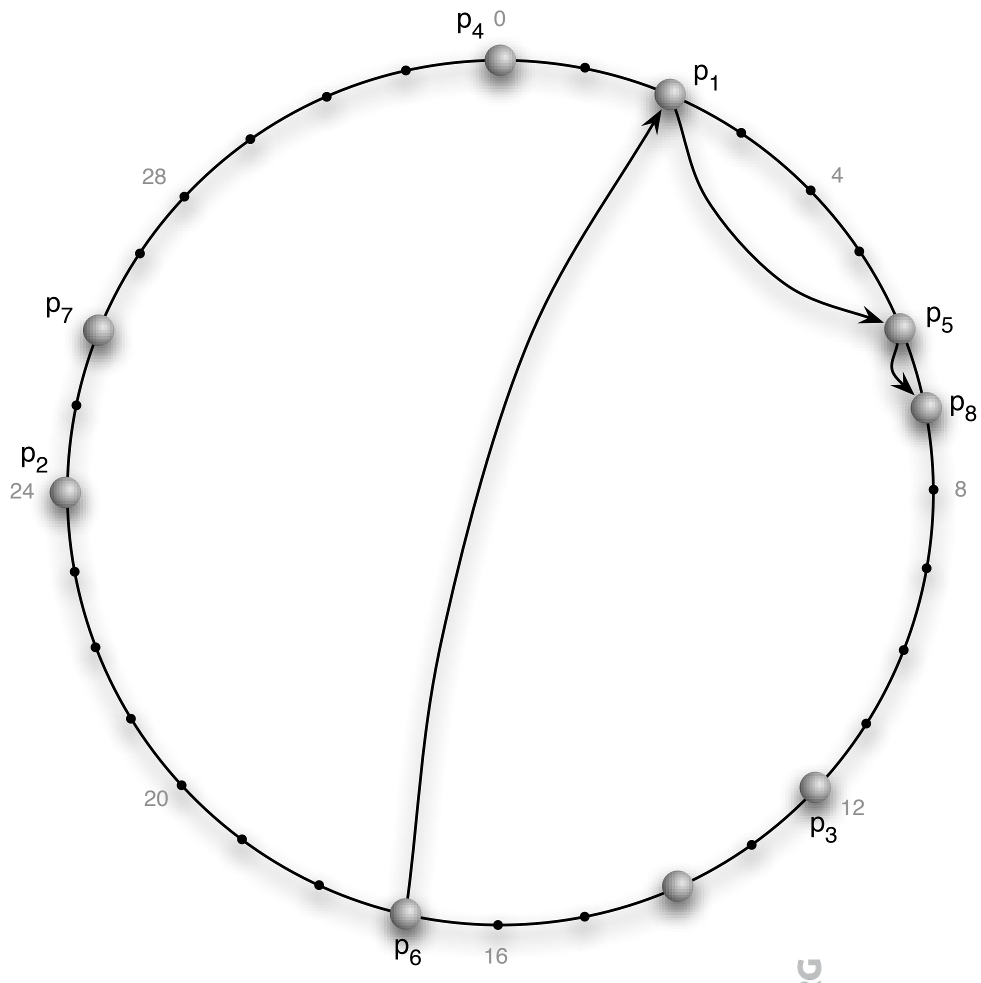
Data Structure of Chord

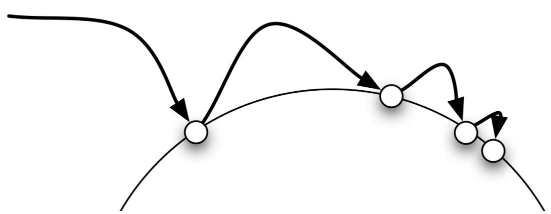
- **For each peer**
  - successor link on the ring
  - predecessor link on the ring
  - for all i ∈ {0,..,m-1}
    - Finger[i] := the peer following the value $r_V(b+2^i)$
- **For small i the finger entries are the same**
  - store only different entries
- **Chord**
  - needs O(log n) hops for lookup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



Lookup in Chord

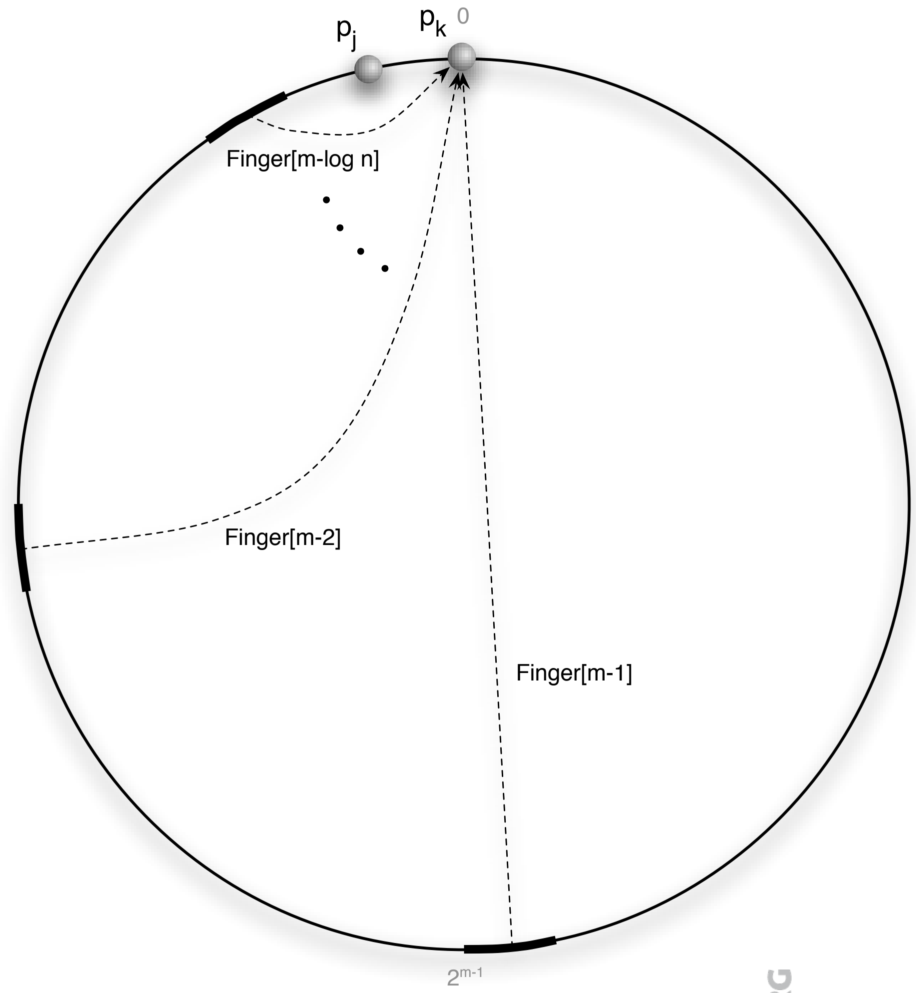
- **Theorem**
  - The Lookup in Chord needs O(log n) steps w.h.p.





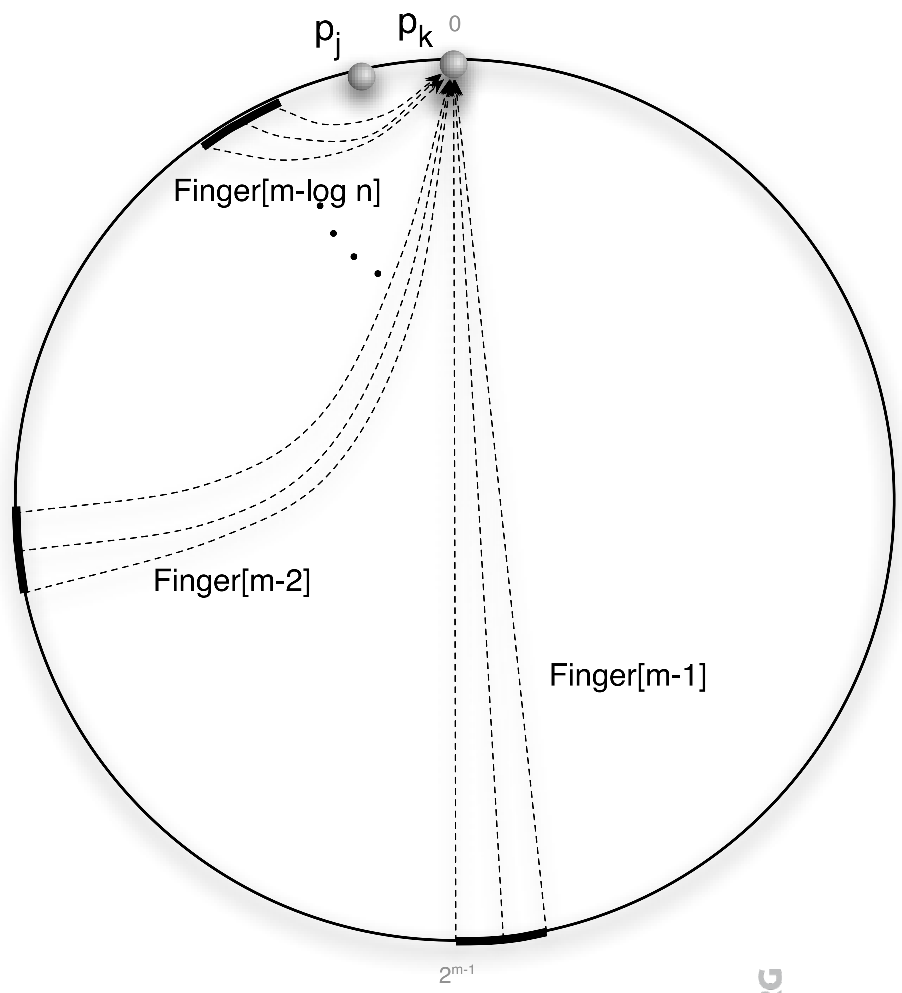
How Many Fingers?

- **Lemma**
  - The out-degree in Chord is O(log n) w.h.p.
  - The in-degree in Chord is O($\log^2$n) w.h.p.
- **Theorem**
  - For integrating a new peer into Chord only O($\log^2$ n) messages are necessary.



Adding a Peer

- **First find the target area in O(log n) steps**
- **The outgoing pointers are adopted from the predecessor and successor**
  - the pointers of at most O(log n) neighbored peers must be adapted
- **The in-degree of the new peer is O ($\log^2 n$) w.h.p.**
  - Lookup time for each of them
  - There are O(log n) groups of neighb ored peers
  - Hence, only O(log n) lookup steps with at most costs O(log n) must be used
  - Each update of has constant cost



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
  - ℓ: 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

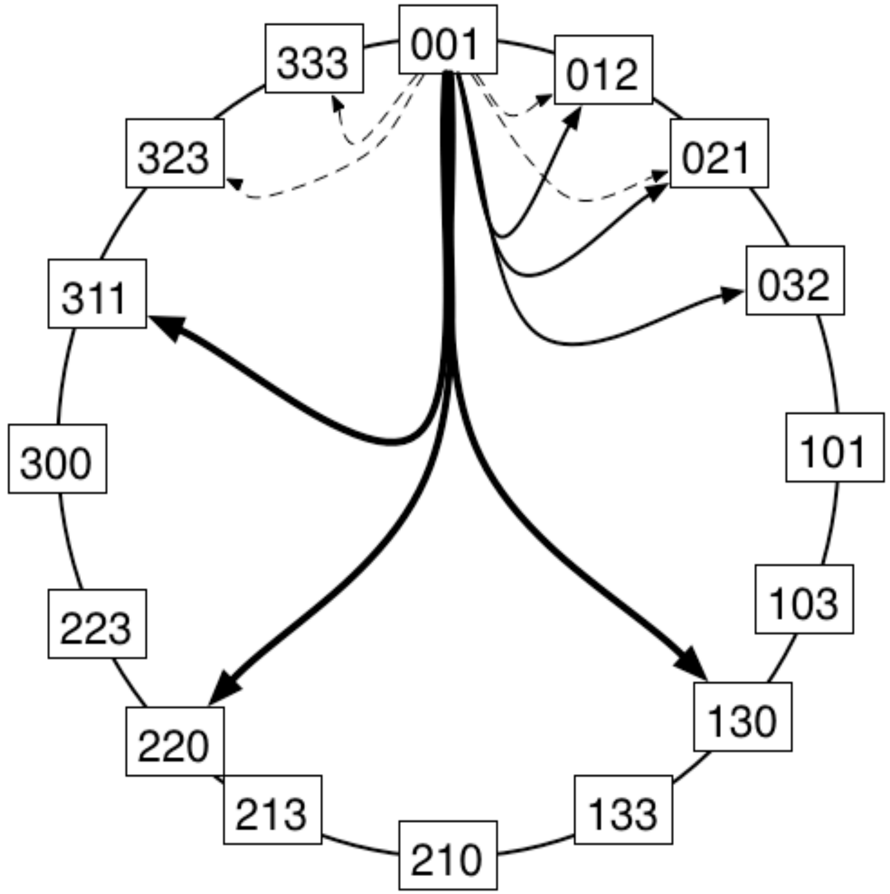
Routing Table

- **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 ℓ neighors**
  - $\ell/2$ 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 neighbors**
  - 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



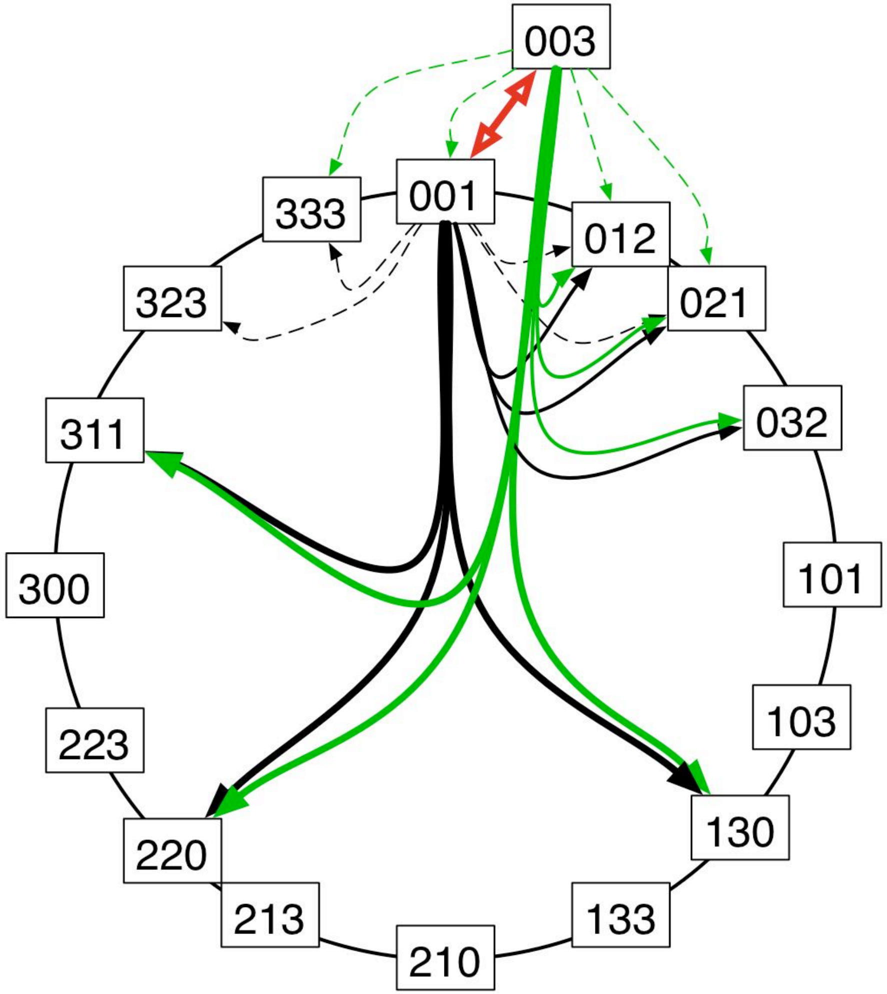
Routing Table

- 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 $$(1-2^{-bm})^n \leq e^{-n/2^{bm}}$$
  - For m=c (log n)/b we get $$e^{-n/2^{bm}} \leq e^{-n/2^{c \log n}}$$ $$\leq e^{-n/n^c} \leq e^{-n^{c-1}}$$
  - With (extremely) high probability there is no peer with the same prefix of length (1+ε)(log n)/b
  - Hence we have (1+ε)(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

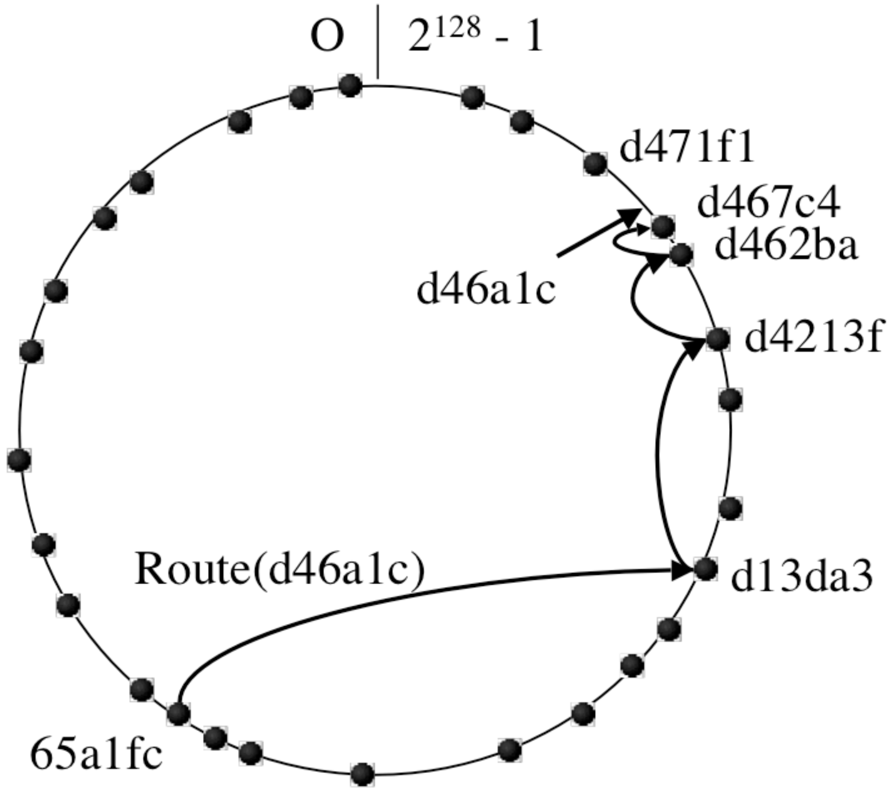
A Peer Enters

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



Lookup

- **Compute the target ID using the hash function**
- **If the address is within the ℓ-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**



Routing — Discussion

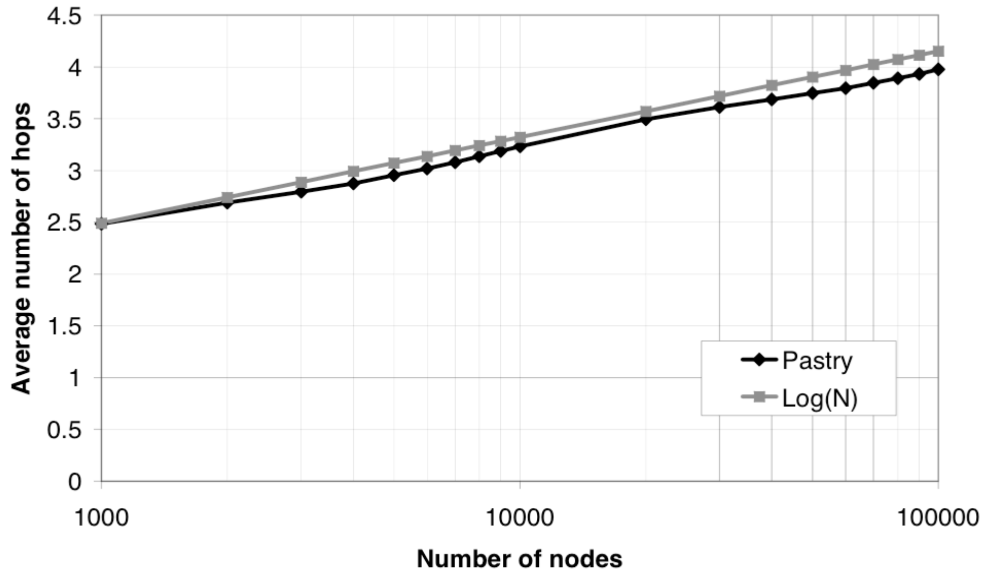
- If the Routing-Table is correct
  - routing needs O((log n)/b) messages
- As long as the leaf-set is correct
  - routing needs O(n/l) messages
  - unrealistic worst case since even damaged routing tables allow dramatic speedup
- Routing does not use the real distances
  - M is used only if errors in 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

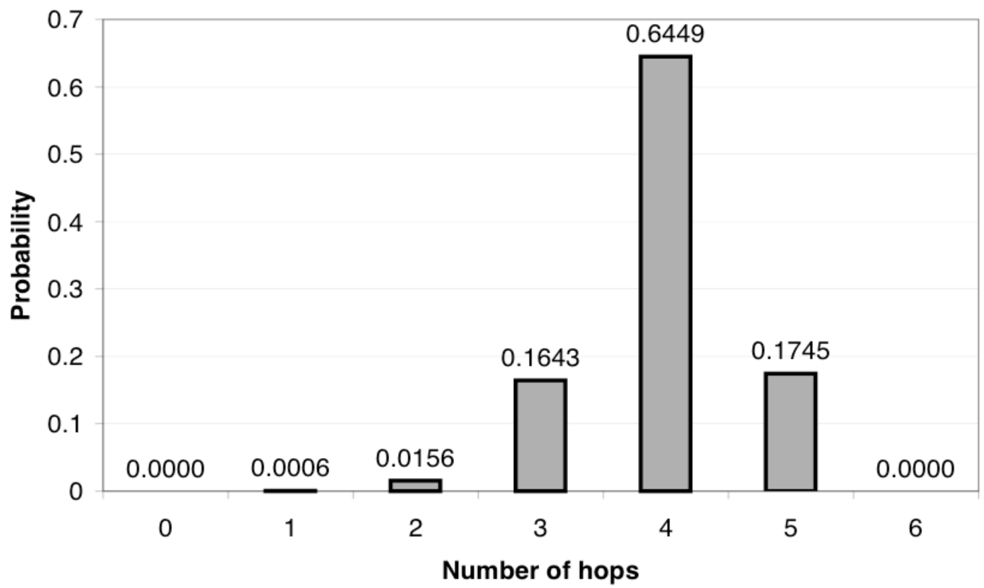
Experimental Results — Scalability

- **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**



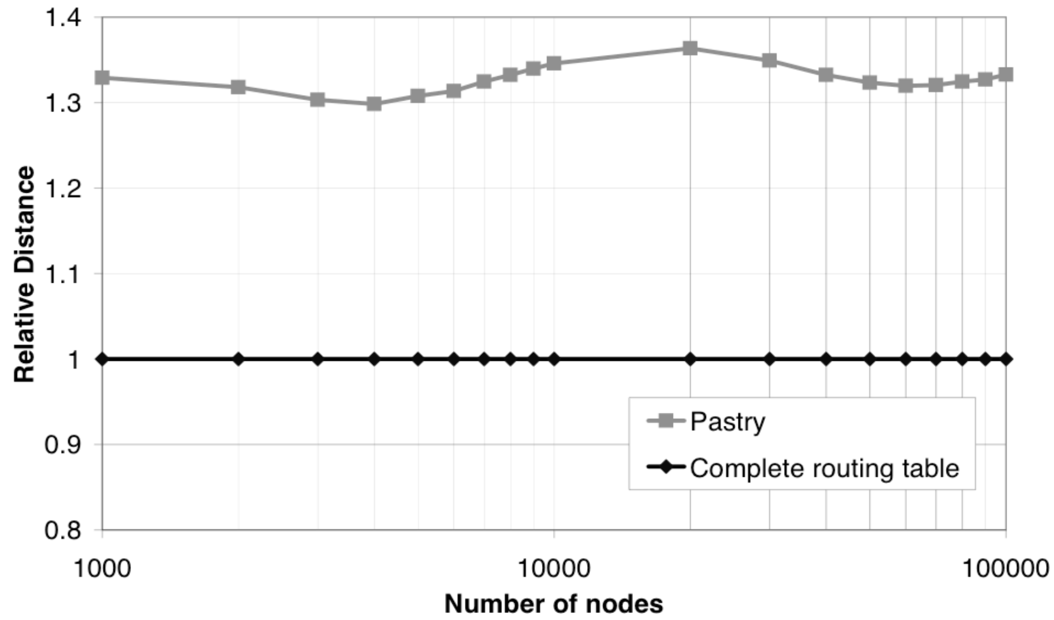
Experimental Results

- **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



DAAD Summerschool Curitiba 2011

Aspects of Large Scale High Speed Computing Building Blocks of a Cloud

Storage Networks

5: Peer-to-Peer Networks

Christian Schindelhauer

Technical Faculty

Computer-Networks and Telematics

University of Freiburg