Distributed Systems

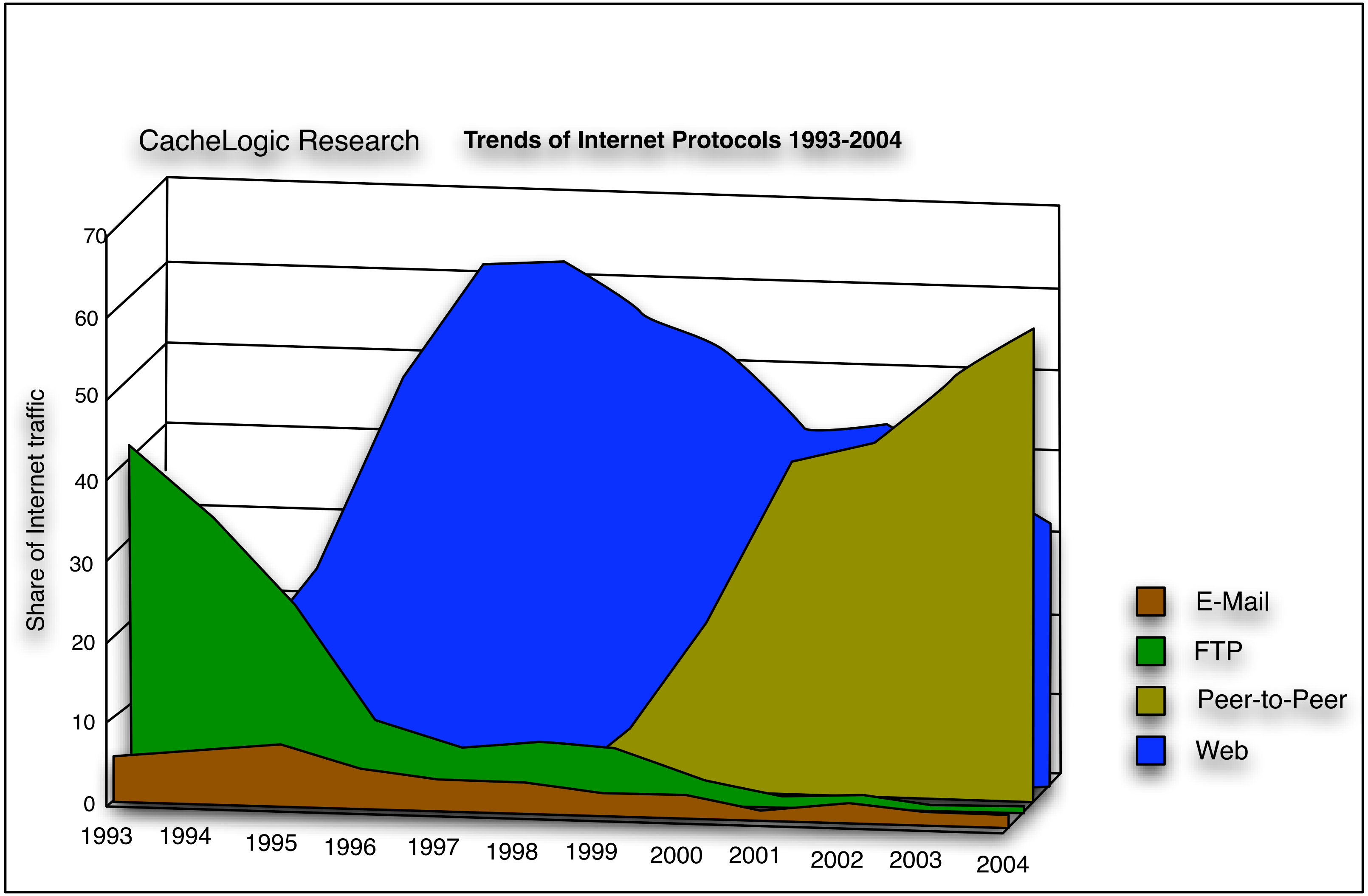
7: Peer-to-Peer-Networks

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg

What is a P2P Network?

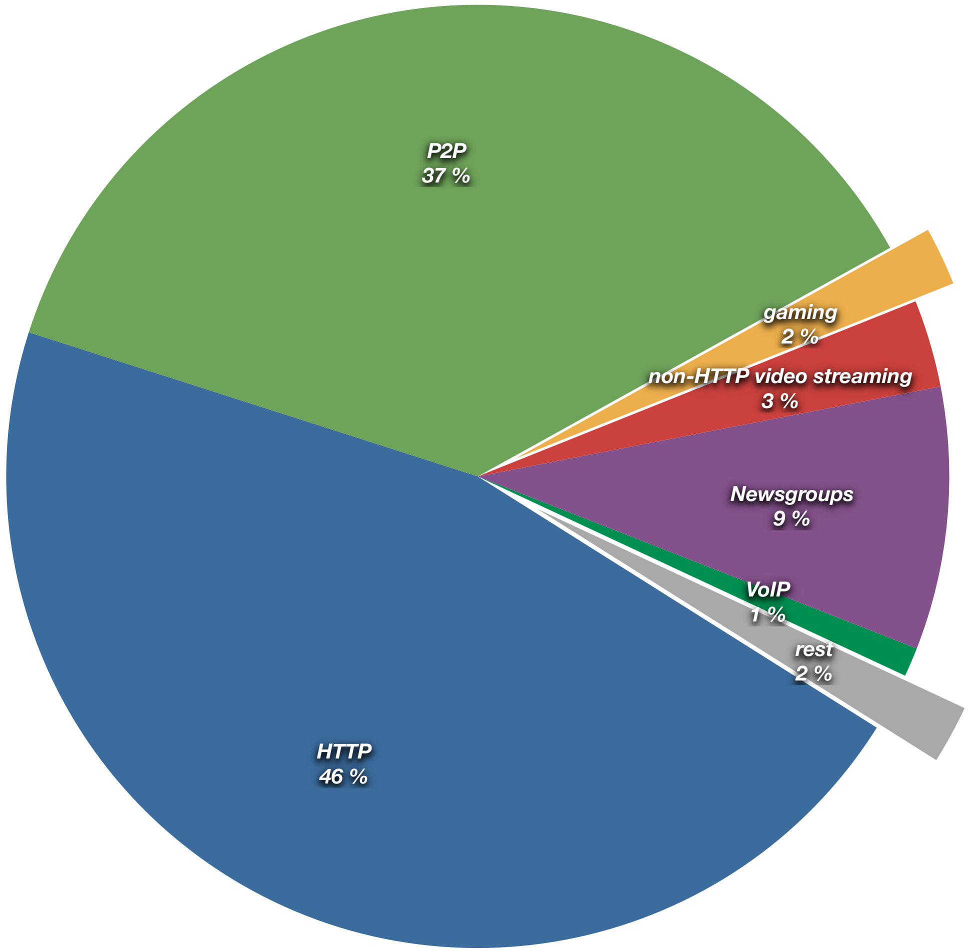
- What is P2P NOT?
  - a peer-to-peer network is not a client-server network
- Etymology: peer
  - from latin par = equal
  - one that is of equal standing with another
  - P2P, Peer-to-Peer: a relationship between equal partners
- Definition
  - a Peer-to-Peer Network is a communication network between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

Global Internet Traffic Shares 1993-2004



Global Internet Traffic 2007

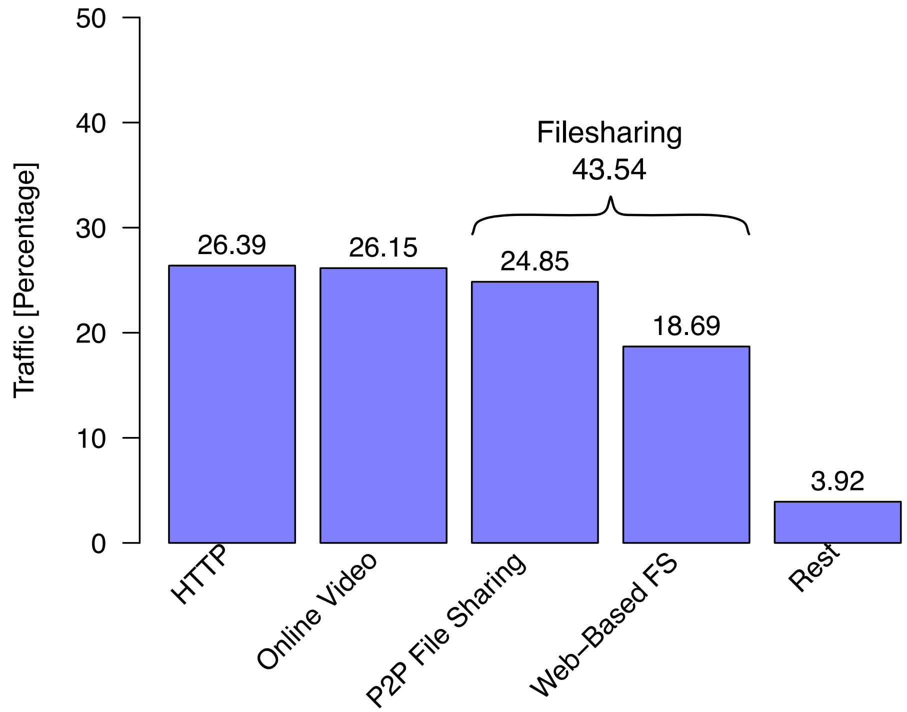
- Ellacoya report (June 2007)
  - worldwide HTTP traffic volume overtakes P2P after four years continues record
- Main reason: Youtube.com



Internet Traffic 2010

- Cisco Visual Networking Index Usage
- contains data of 20 anonymous service providers

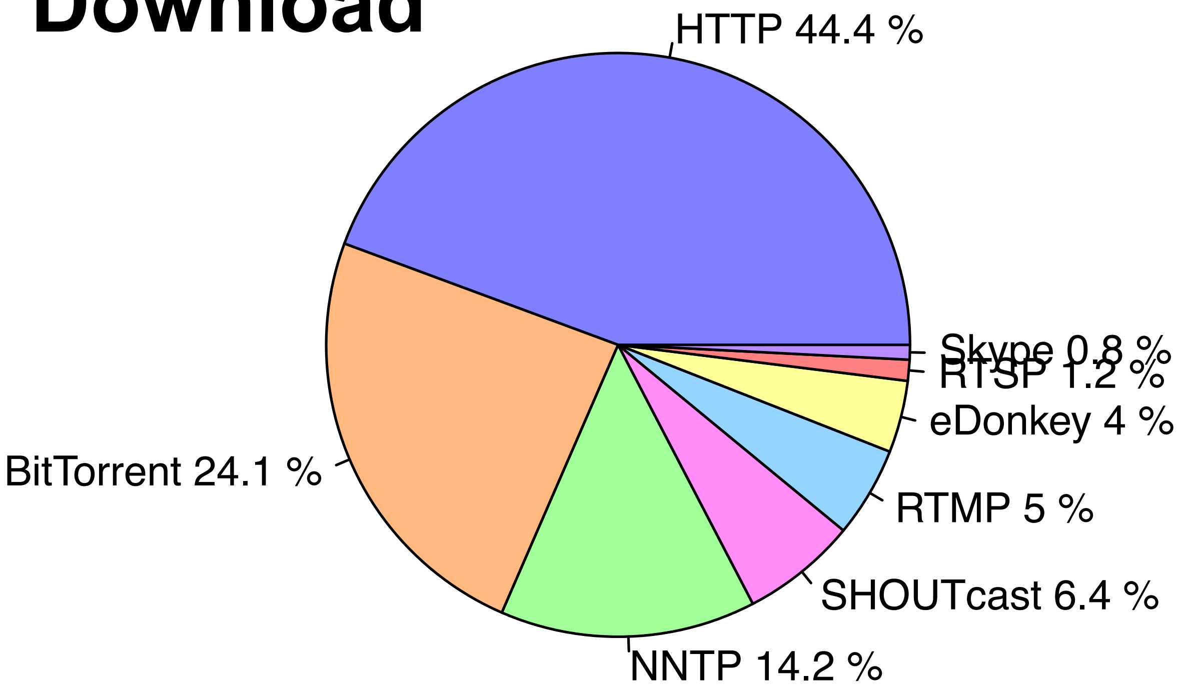
Traffic Study



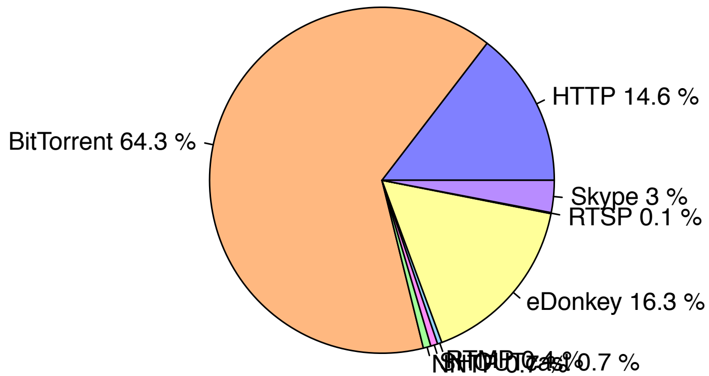
["Cisco Visual Networking Index: Usage", White Paper, 2010]

Internet Traffic of a German ISP August 2009

Download



Upload

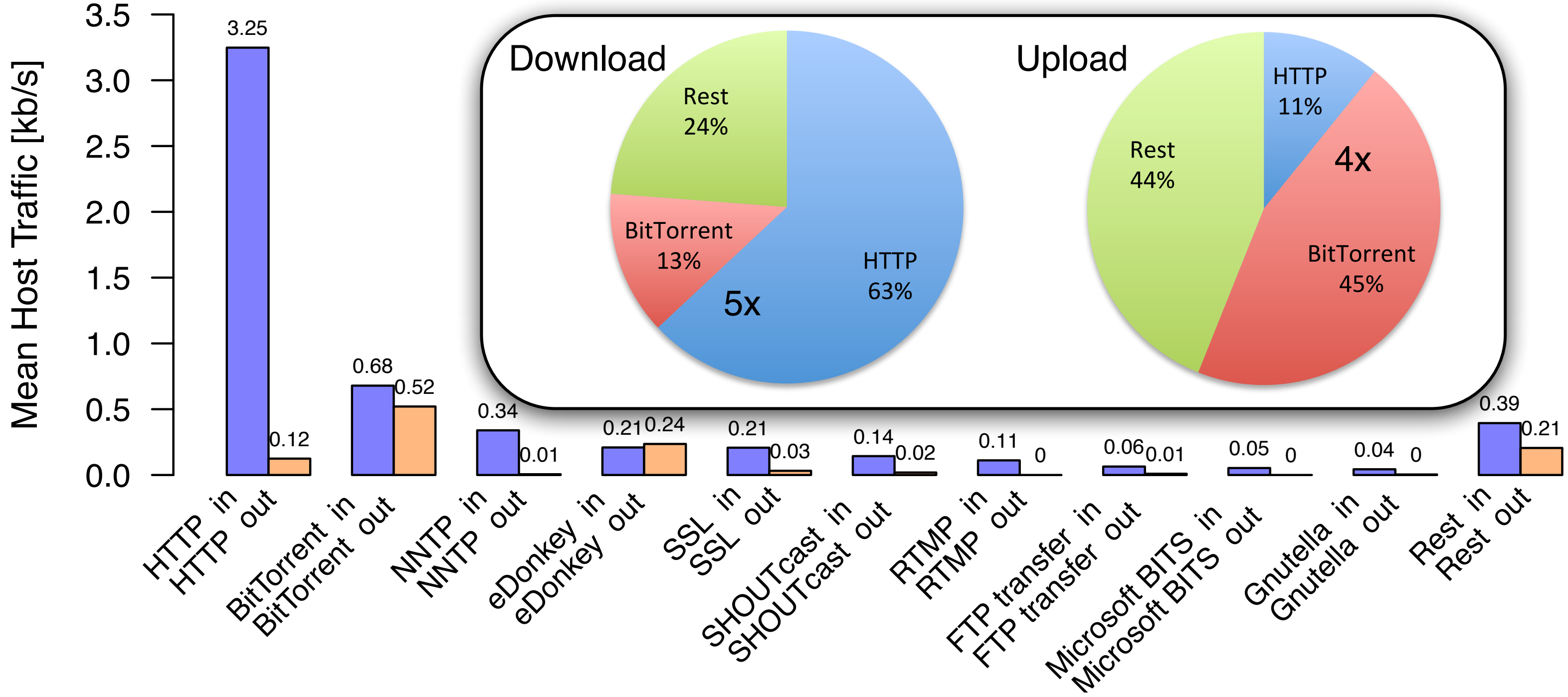


Source: Alsbih, Janson, S. Analysis of Peer-to-Peer Traffic and User Behaviour ITA 2011

Internet Traffic of a German ISP August 2009

- HTTP most traffic
- BitTorrent most upload

Top ten services of the average user



Source: Alsbih, Janson, S. Analysis of Peer-to-Peer Traffic and User Behaviour ITA 2011

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

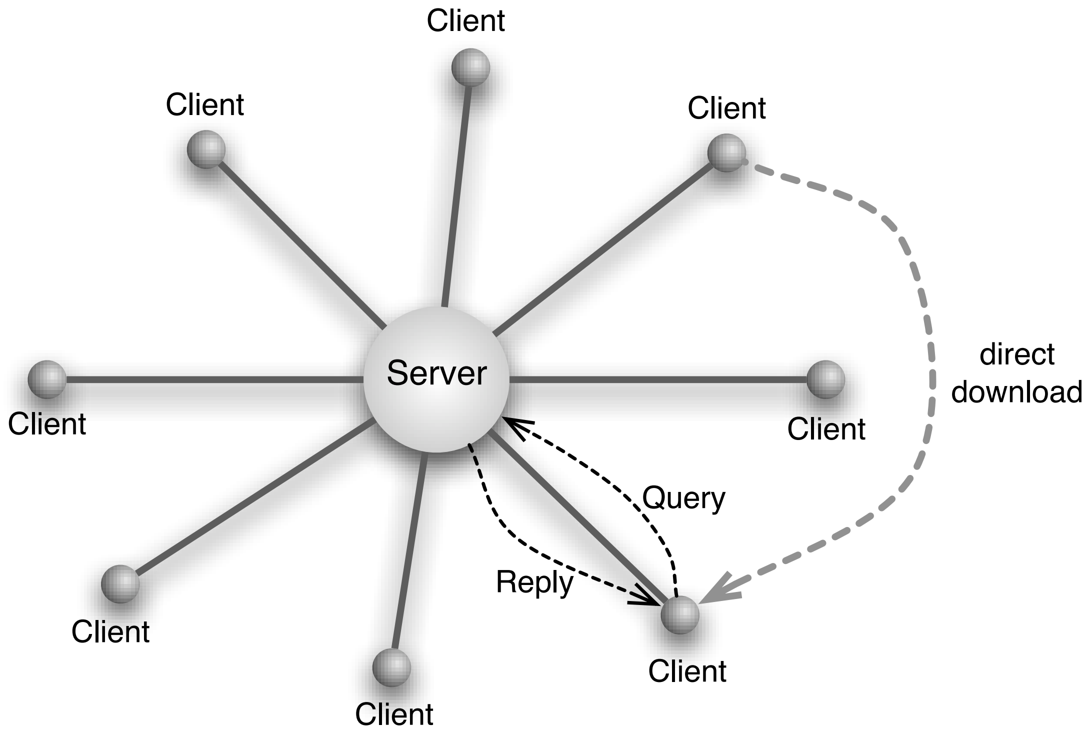
- Distributed Hash-Tables (DHT) (1997)
  - introduced for load balancing between web-servers
- CAN (2001)
  - efficient distributed DHT data structure for P2P networks
- Chord (2001)
  - efficient distributed P2P network with logarithmic search time
- Pastry/Tapestry (2001)
  - efficient distributed 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

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

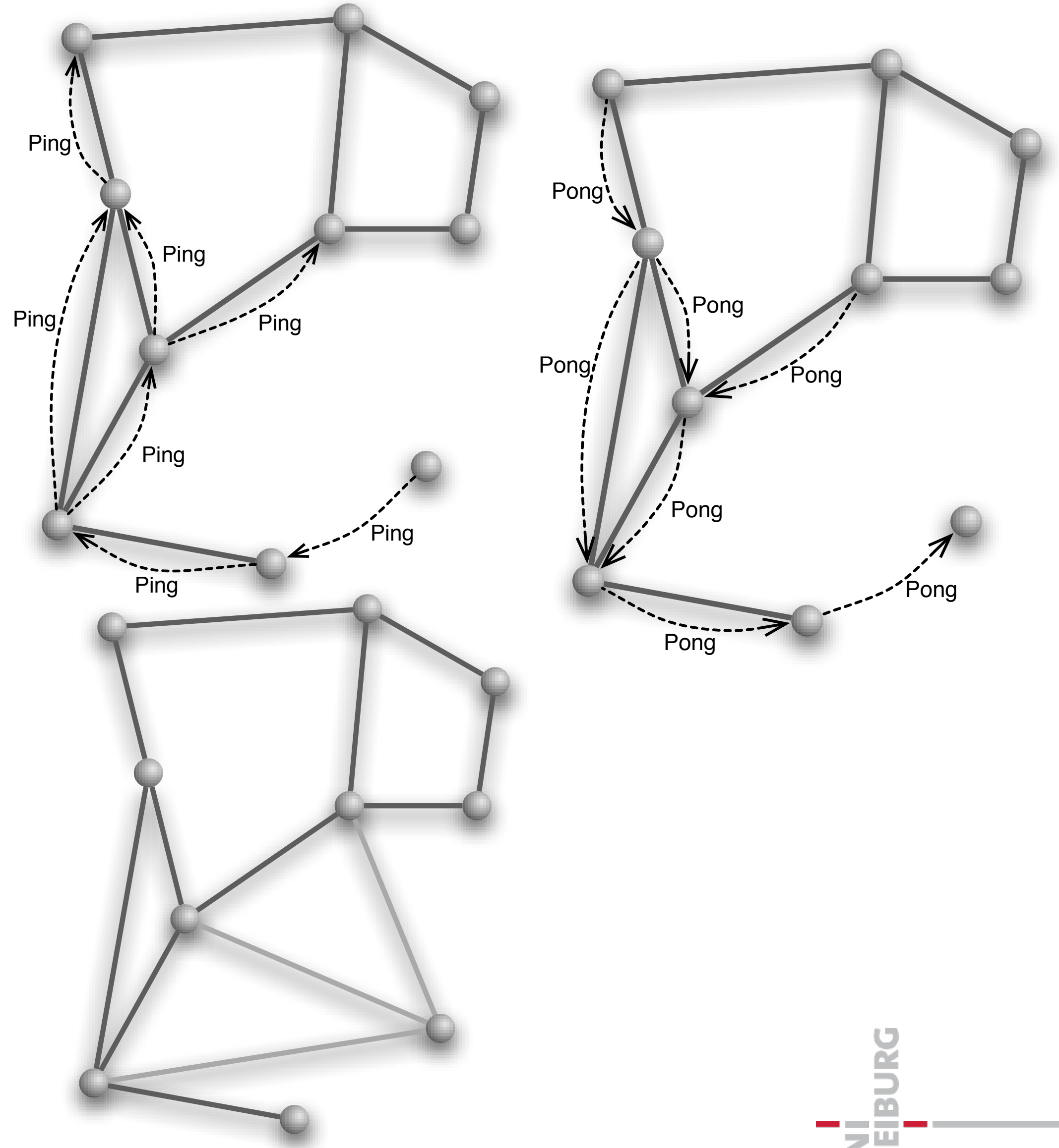


History of Gnutella

- 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

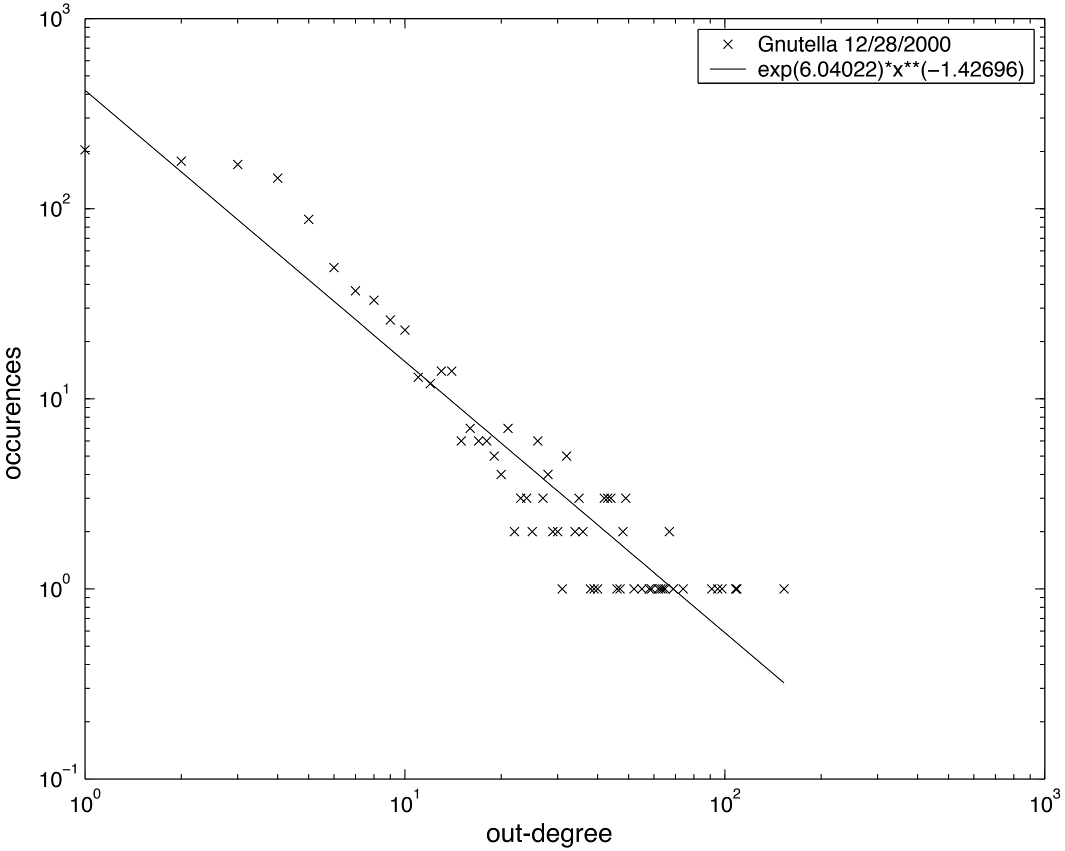
Gnutella — Connecting

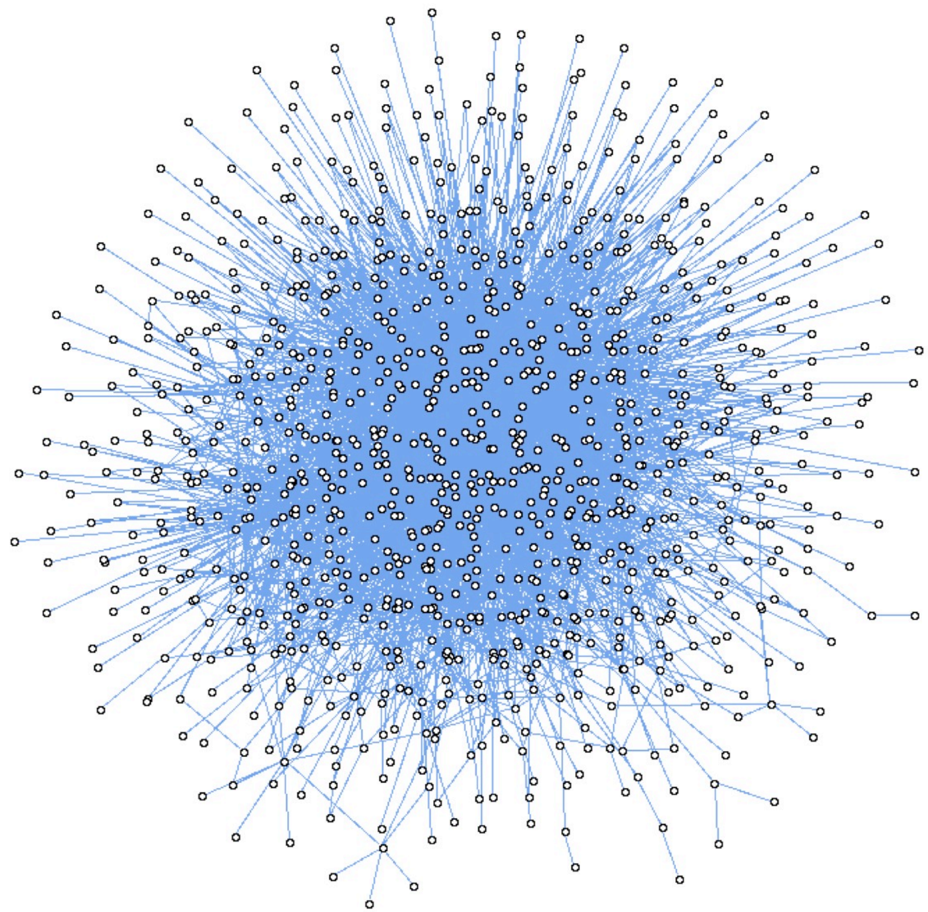
- Neighbor lists
  - Gnutella connects directly with other clients
  - the client software includes a list of usually online clients
  - the clients checks these clients until an active node has been found
  - an active client publishes its neighbor list
  - the query (ping) is forwarded to other nodes
  - 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

Pastry

- Peer-to-Peer Networks

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

CoNe Freiburg

Routing Table

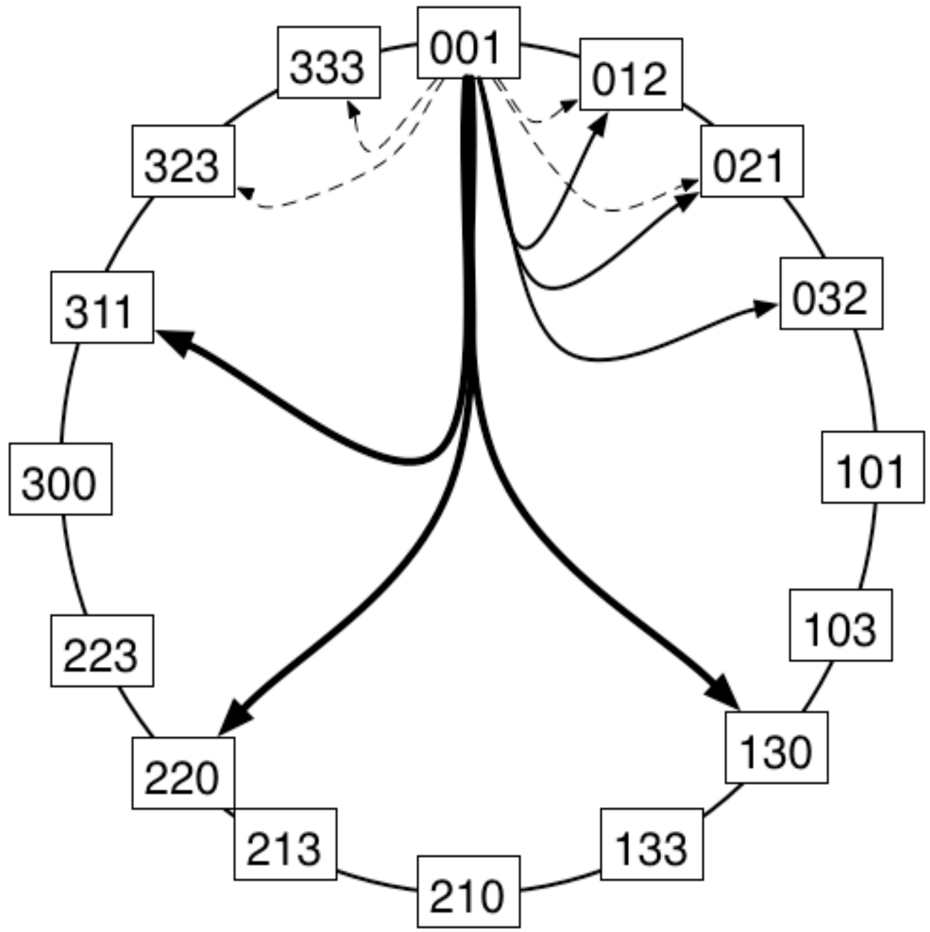
- NodeId presented in base 2^b
  - e.g. NodeID: 65A0BA13
- For each prefix p and letter x ∈ {0,..,2^b-1} add an peer of form px* to the routing table of NodeID, e.g.
  - b=4, 2^b=16
  - 15 entries for 0*,1*, .. F*
  - 15 entries for 60*, 61*,... 6F*
  - ...
  - if no peer of the form exists, then the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

UNI FREIBURG

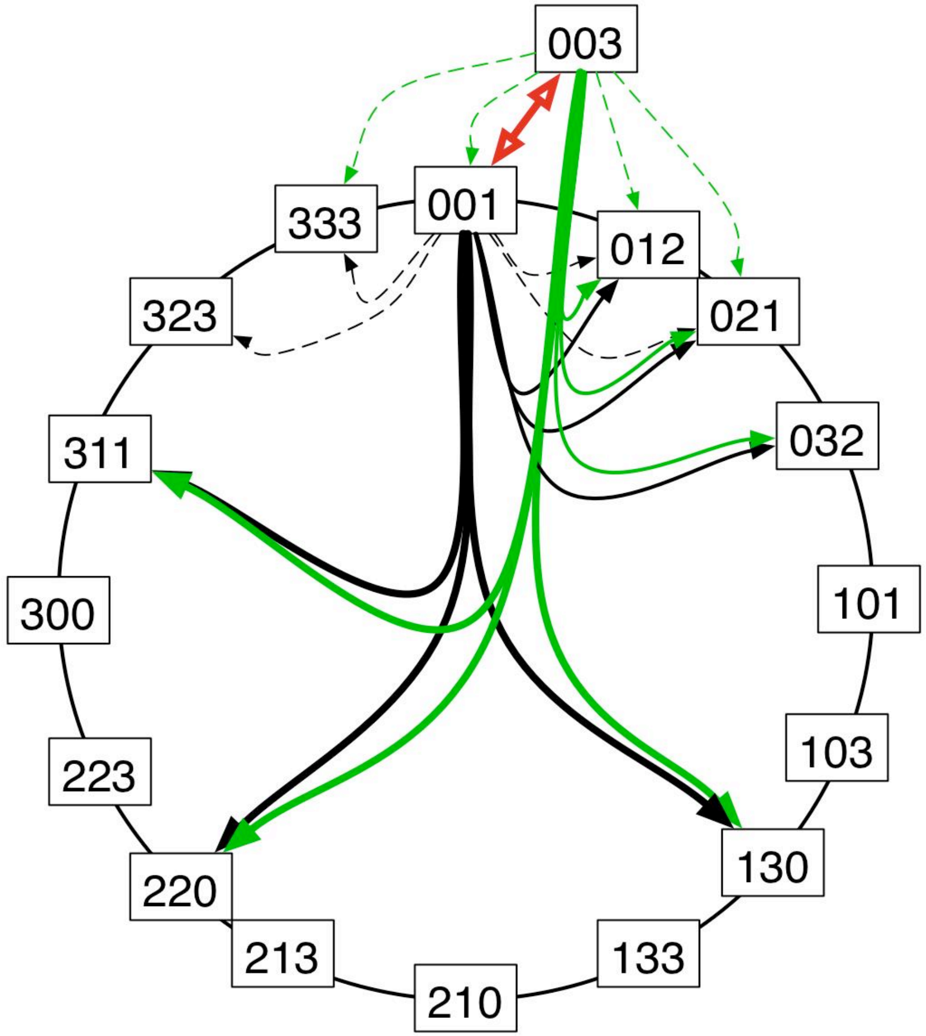
Routing Table

- Example b=2
- Routing Table
  - For each prefix p and letter x ∈
    form px* to the routing table of
- 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
- Theorem
  - With high probability there are
    entries in each routing table



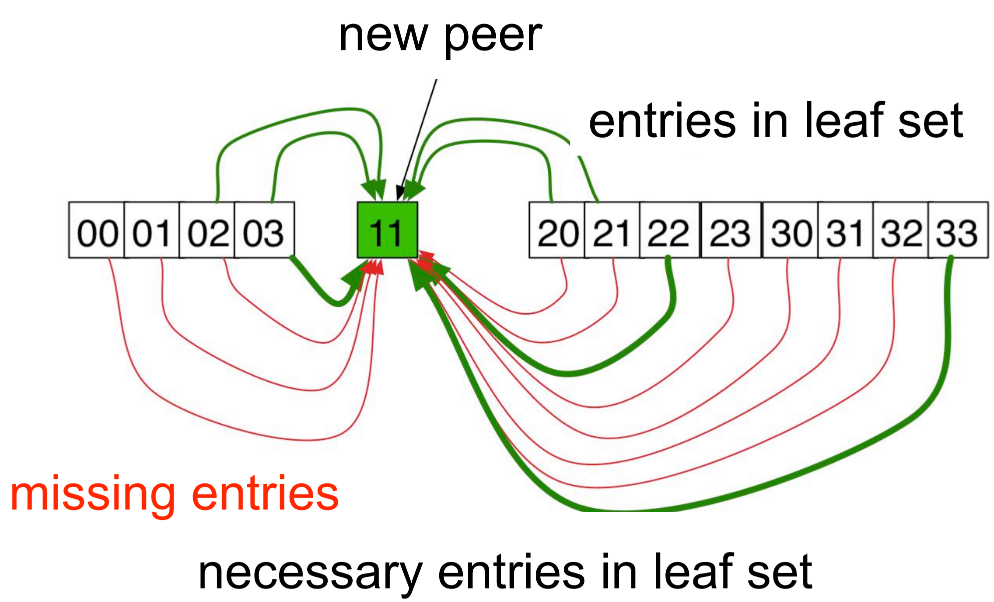
A Peer Enters

- 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



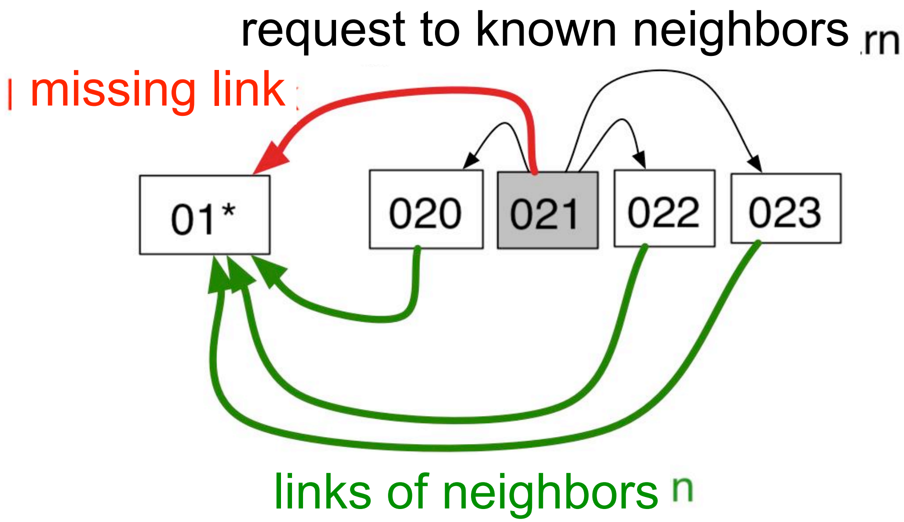
When the Entry-Operation Errs

- 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



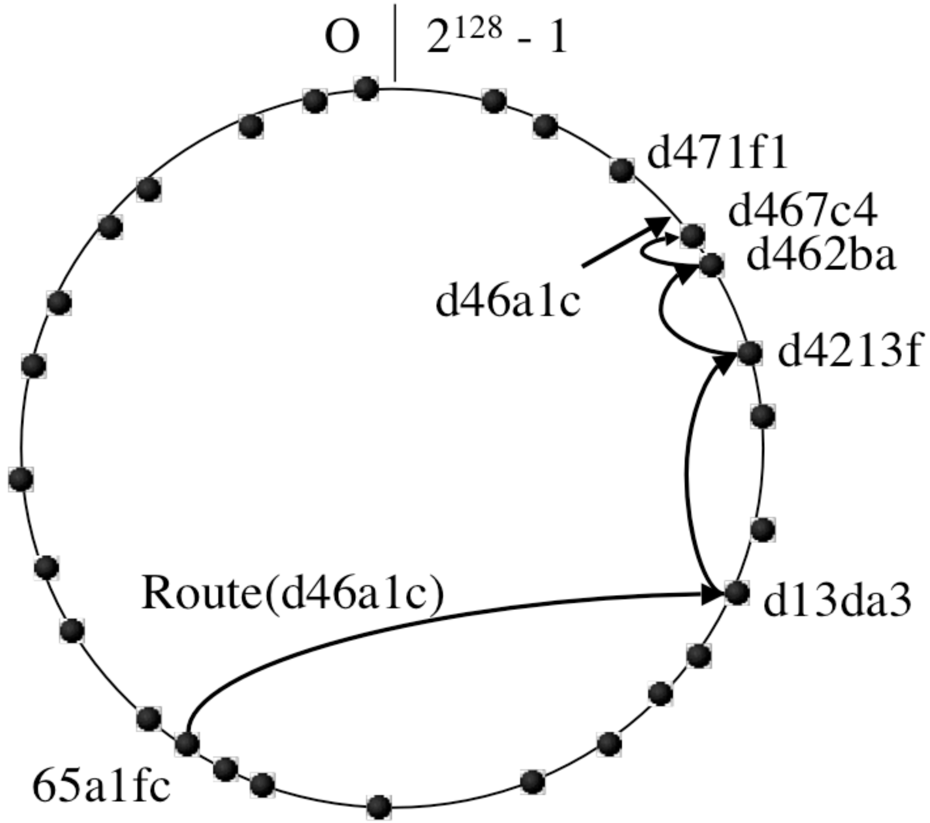
Missing Entries in the Routing Table

- 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 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



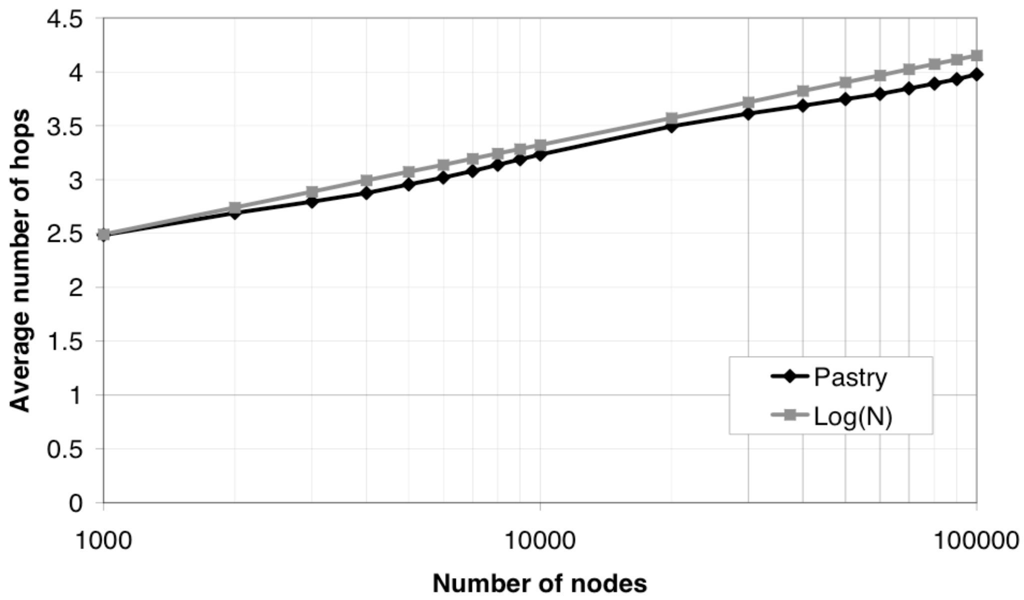
CoNe Freiburg

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

Experimental Results — Scalability

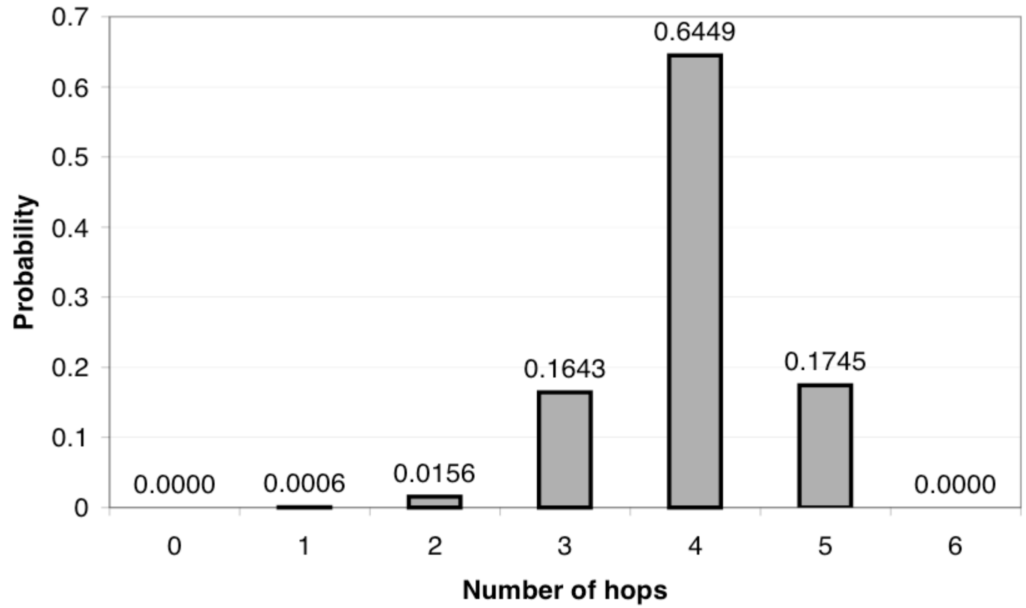
- Parameter b=4, l=16, M=32
- In this experiment the hop distance grows logarithmically with the number of nodes
- The analysis predicts O(log n)
- Fits well



CoNe Freiburg

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



UNI FREIBURG

Past by Druschel, Rowstron 2001

- Distributed Storage

PAST

- PAST: A large-scale, persistent peer-to-peer storage utility
  - by Peter Druschel (Rice University, Houston – now Max-Planck-Institut, Saarbrücken/Kaiserlautern)
  - and Antony Rowstron (Microsoft Research)
- Literature
  - A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", 18th ACM SOSP'01, 2001.
    - all pictures from this paper
  - P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-to-peer storage utility", HotOS VIII, May 2001.

Goals of PAST

- Peer-to-Peer based Internet Storage
  - on top of Pastry
- Goals
  - File based storage
  - High availability of data
  - Persistent storage
  - Scalability
  - Efficient usage of resources

Motivation

- Multiple, diverse nodes in the Internet can be used
  - safety by different locations
- No complicated backup
  - No additional backup devices
  - No mirroring
  - No RAID or SAN systems with special hardware
- Joint use of storage
  - for sharing files
  - for publishing documents
- Overcome local storage and data safety limitations

Interface of PAST

- Create:
  `fileId = Insert(name, owner-credentials, k, file)`
  - stores a file at a user-specified number k of divers nodes within the PAST network
  - produces a 160 bit ID which identifies the file (via SHA-1)
- Lookup:
  `file = Lookup(fileId)`
  - reliably retrieves a copy of the file identified fileId
- Reclaim:
  `Reclaim(fileId, owner-credentials)`
  - reclaims the storage occupied by the k copies of the file identified by fileId

- Other operations do not exist:
  - No erase
    - to avoid complex agreement protocols
  - No write or rename
    - to avoid write conflicts
  - No group right management
    - to avoid user, group managements
  - No list files, file information, etc.
- Such operations must be provided by additional layer

Relevant Parts of Pastry

- Leafset:
  - Neighbors on the ring
- Routing Table
  - Nodes for each prefix + 1 other letter
- Neighborhood set
  - set of nodes which have small TTL

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

Interfaces of Pastry

- route(M, X):
  - route message M to node with nodeId numerically closest to X
- deliver(M):
  - deliver message M to application
- forwarding(M, X):
  - message M is being forwarded towards key X
- newLeaf(L):
  - report change in leaf set L to application

Insert Request Operation

- Compute fileId by hashing
  - file name
  - public key of client
  - some random numbers, called salt
- Storage (k x filesize)
  - is debited against client‘s quota
- File certificate
  - is produced and signed with owner‘s private key
  - contains fileID, SHA-1 hash of file‘s content, replciation factor k, the random salt, creation date, etc.
- File and certificate are routed via Pastry
  - to node responsible for fileID
- When it arrives in one node of the k nodes close to the fileId
  - the node checks the validityof the file
  - it is duplicated to all other k-1 nodes numerically close to fileId
- When all k nodes have accepted a copy
  - Each nodes sends store receipt is send to the owner
- If something goes wrong an error message is sent back
  - and nothing stored

Lookup

- Client sends message with requested fileId into the Pastry network
- The first node storing the file answers
  - no further routing
- The node sends back the file
- Locality property of Pastry helps to send a close-by copy of a file

Reclaim

- Client‘s nodes sends reclaim certificate
  - allowing the storing nodes to check that the claim is authentificated
- Each node sends a reclaim receipt
- The client sends this recept to the retrieve the storage from the quota management

Security

- Smartcard
  - for PAST users which want to store files
  - generates and verifies all certificates
  - maintain the storage quotas
  - ensure the integrity of nodeID and fileID assignment
- Users/nodes without smartcard
  - can read and serve as storage servers
- Randomized routing
  - prevents intersection of messages
- Malicious nodes only have local influence

Storage Management

- Goals
  - Utilization of all storage
  - Storage balancing
  - Providing k file replicas
- Methods
  - Replica diversion
    - exception to storing replicas nodes in the leafset
  - File diversion
    - if the local nodes are full all replicas are stored at different locations

Summary

- PAST provides a distributed storage system
  - which allows full storage usage and locality features
- Storage management
  - based ond Smartcard system
    - provides a hardware restriction
  - utilization moderately increases failure rates and time behavior

Distributed Systems
7: Peer-to-Peer-Networks

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg