ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

CoNe
Freiburg

Algorithms and Methods for Peer-to-Peer Networks

Tutorial

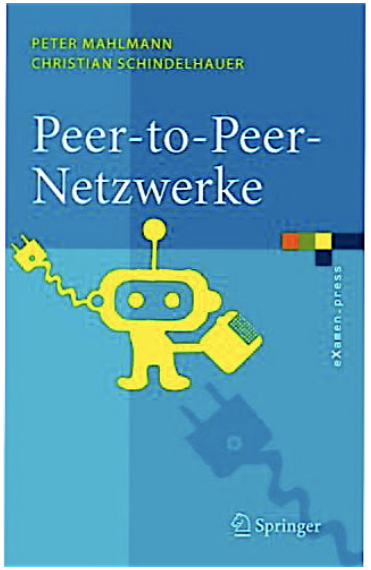
Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Computer Networks and Telematics
Christian Schindelhauer

Overview

- **Motivation & short history**
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Materials

- **Slides**
  - mostly from my lectures
  - http://cone.informatik.uni-freiburg.de/past-teaching-e.html
- **Book**
  - Mahlmann, Schindelhauer, Peer-to-Peer-Netzwerke — Methoden und Algorithmen, Springer 2007
- **Further Literature**
  - Research papers are referenced on the slides



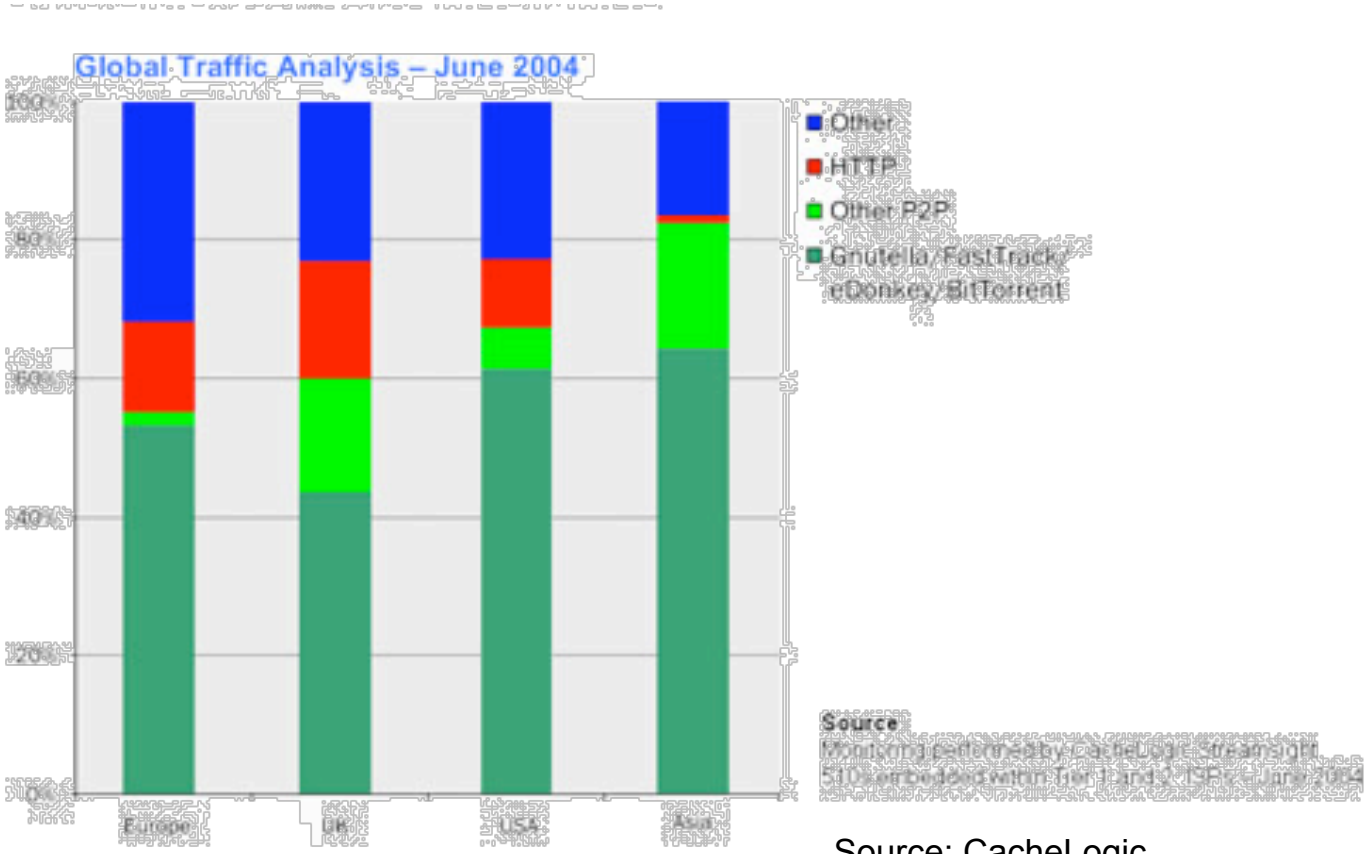
Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Peer-to-Peer Networks

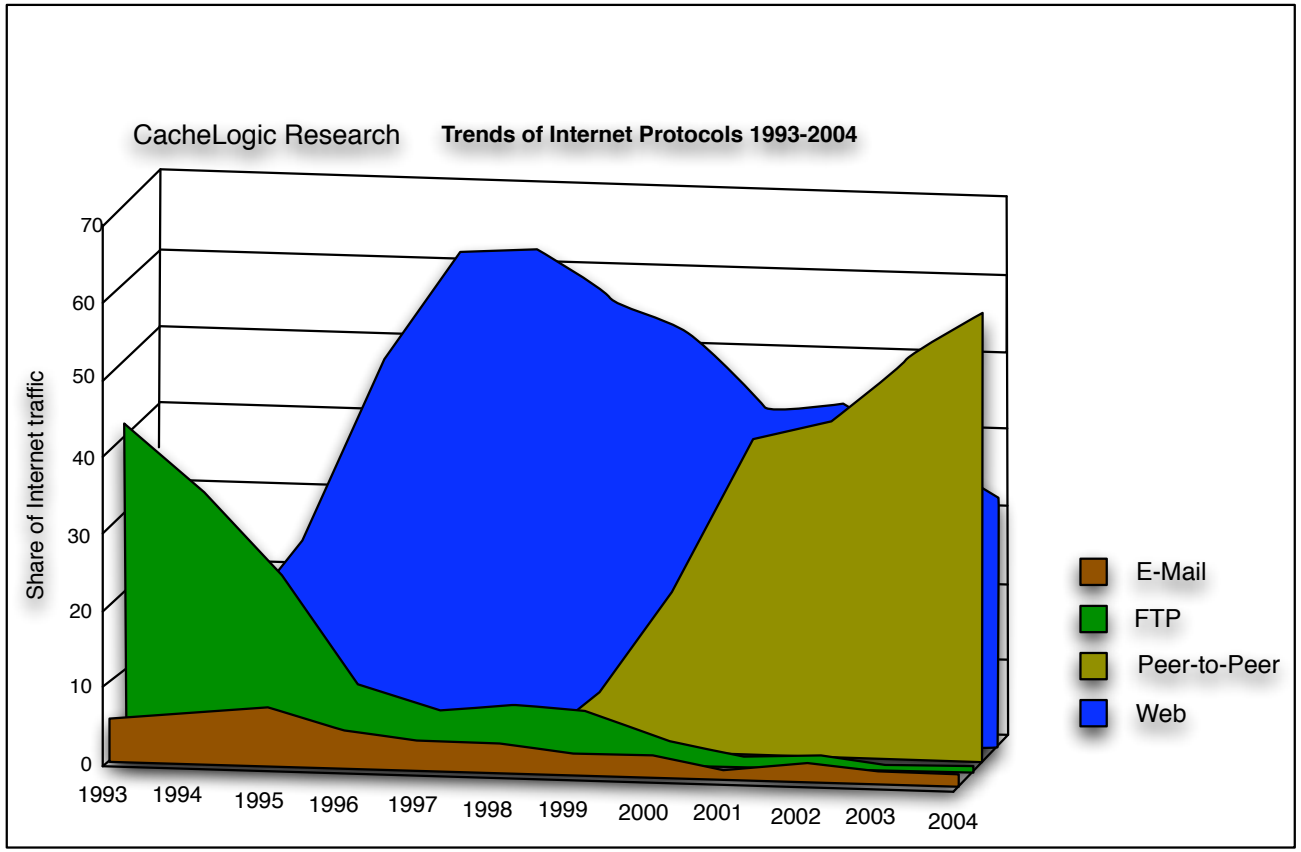
Motivation

P2P Share
June 2004



Source: CacheLogic

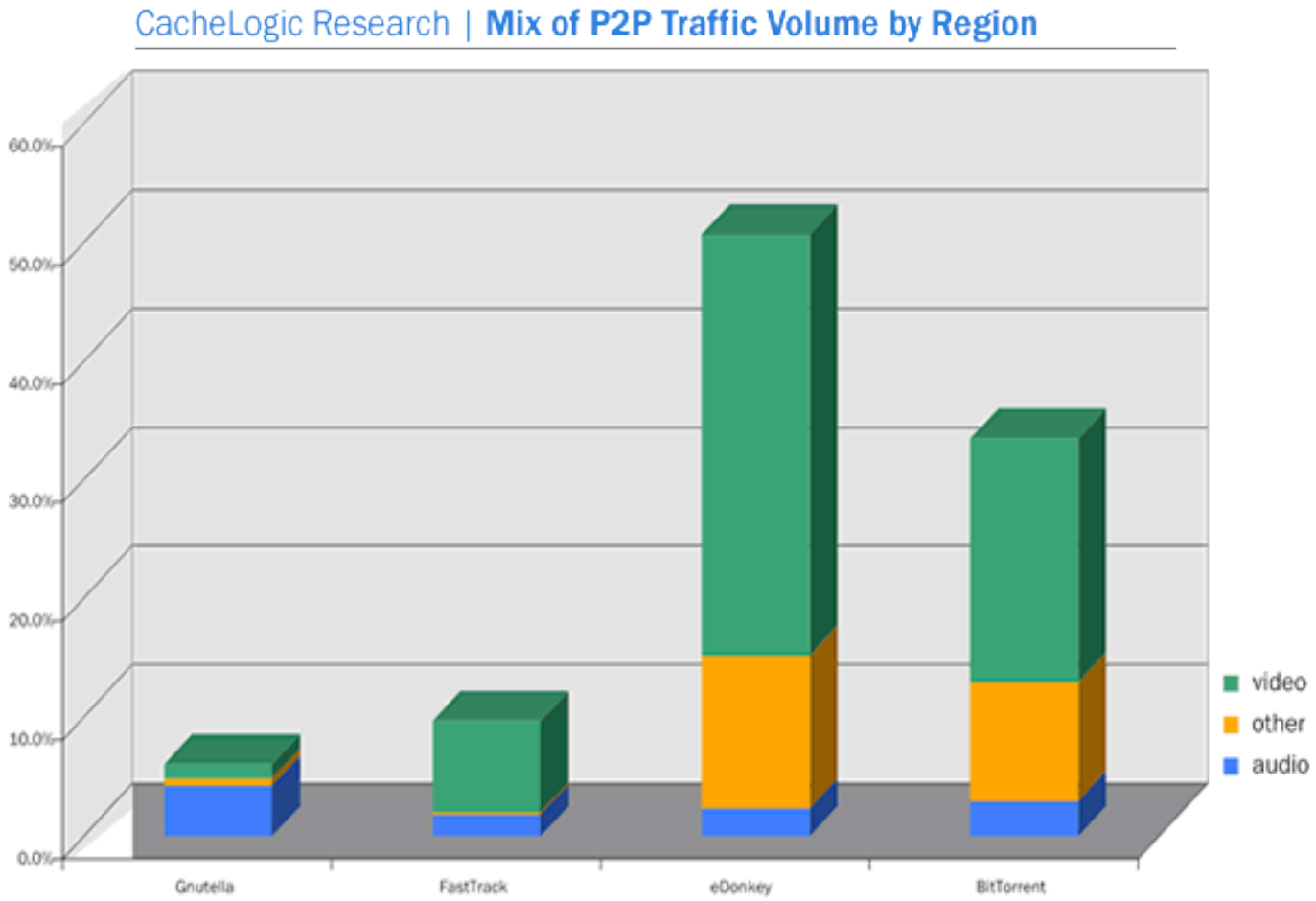
Global Internet Traffic Shares 1993-2004



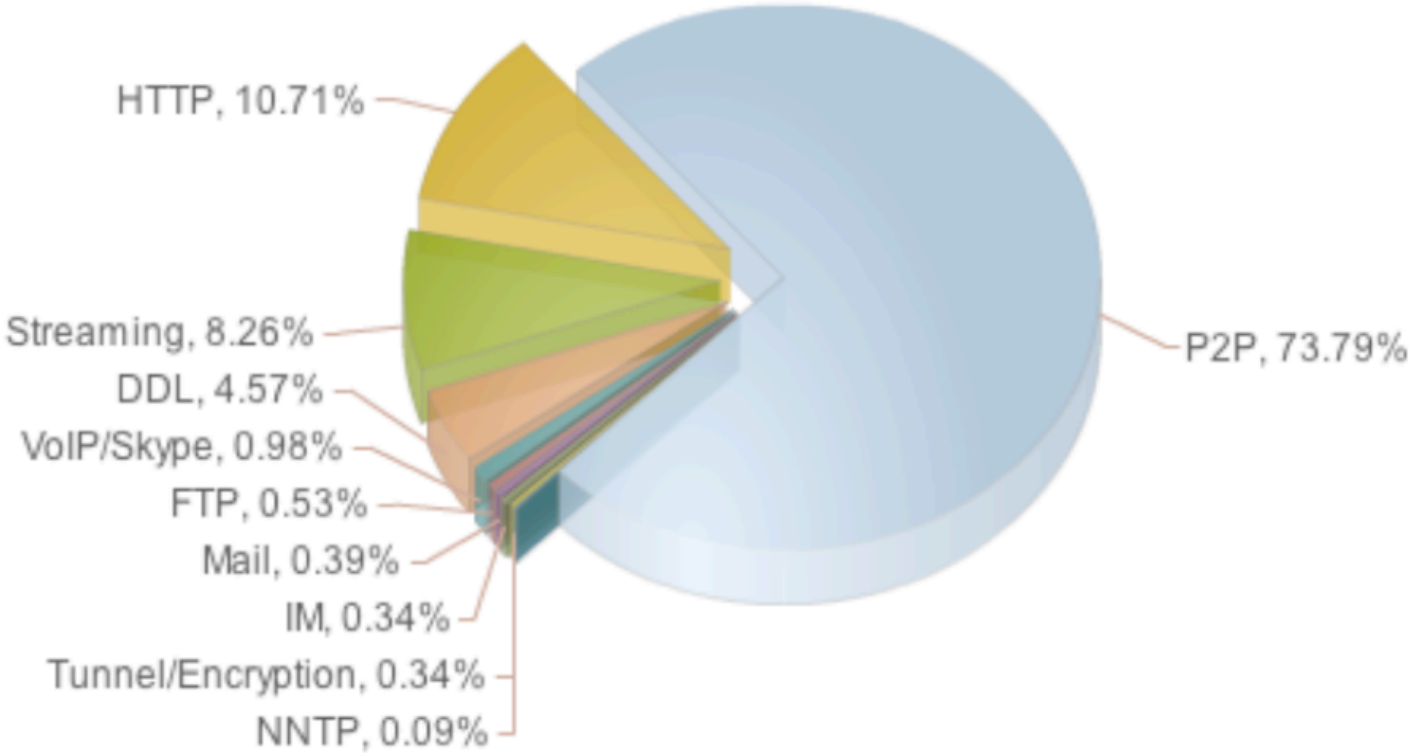
Source: CacheLogic 2005

Main Protocols 2004

- **Main protocols**
  - eDonkey
  - BitTorrent
  - FastTrack
  - Gnutella

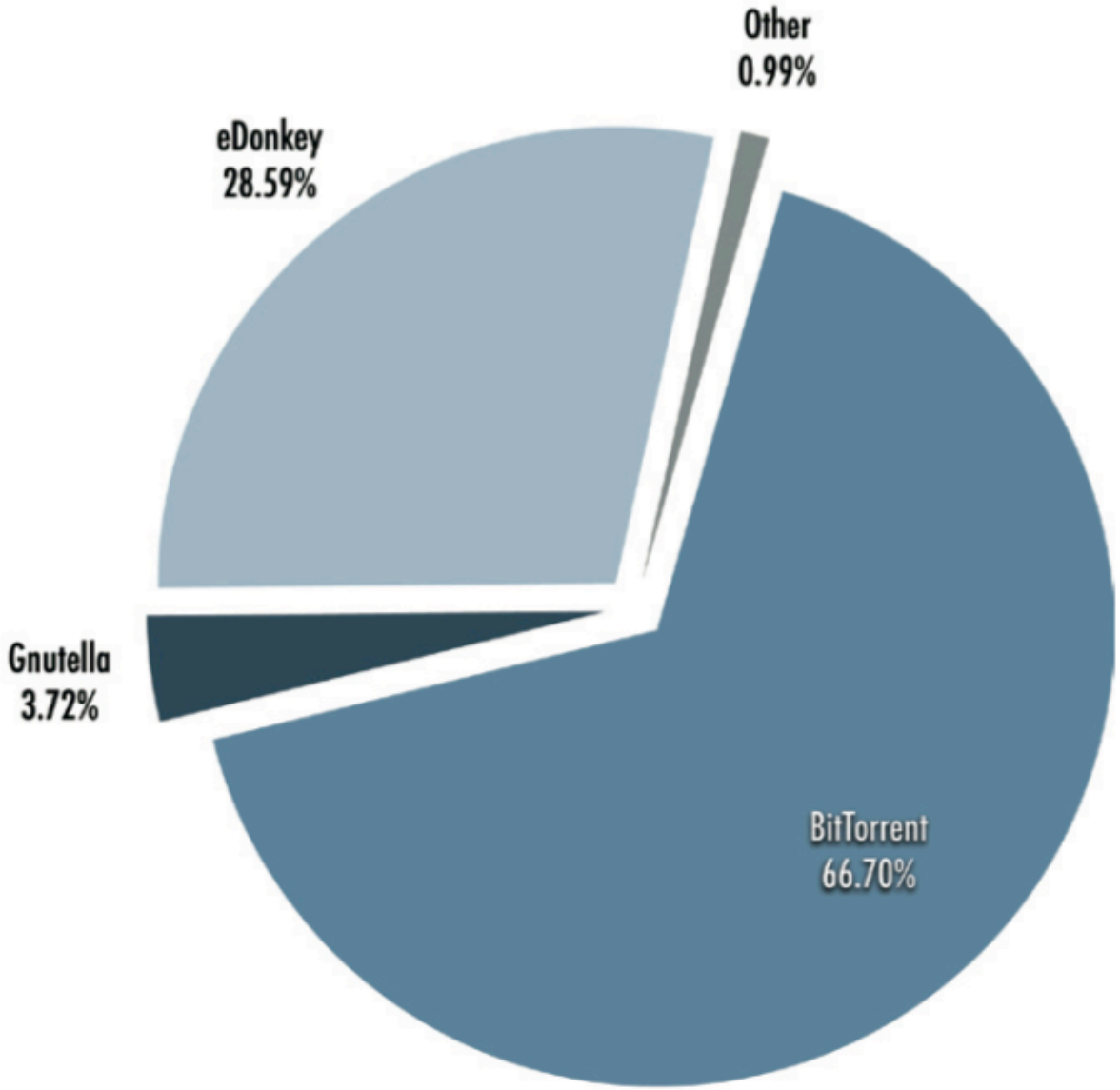


P2P Share Germany 2007



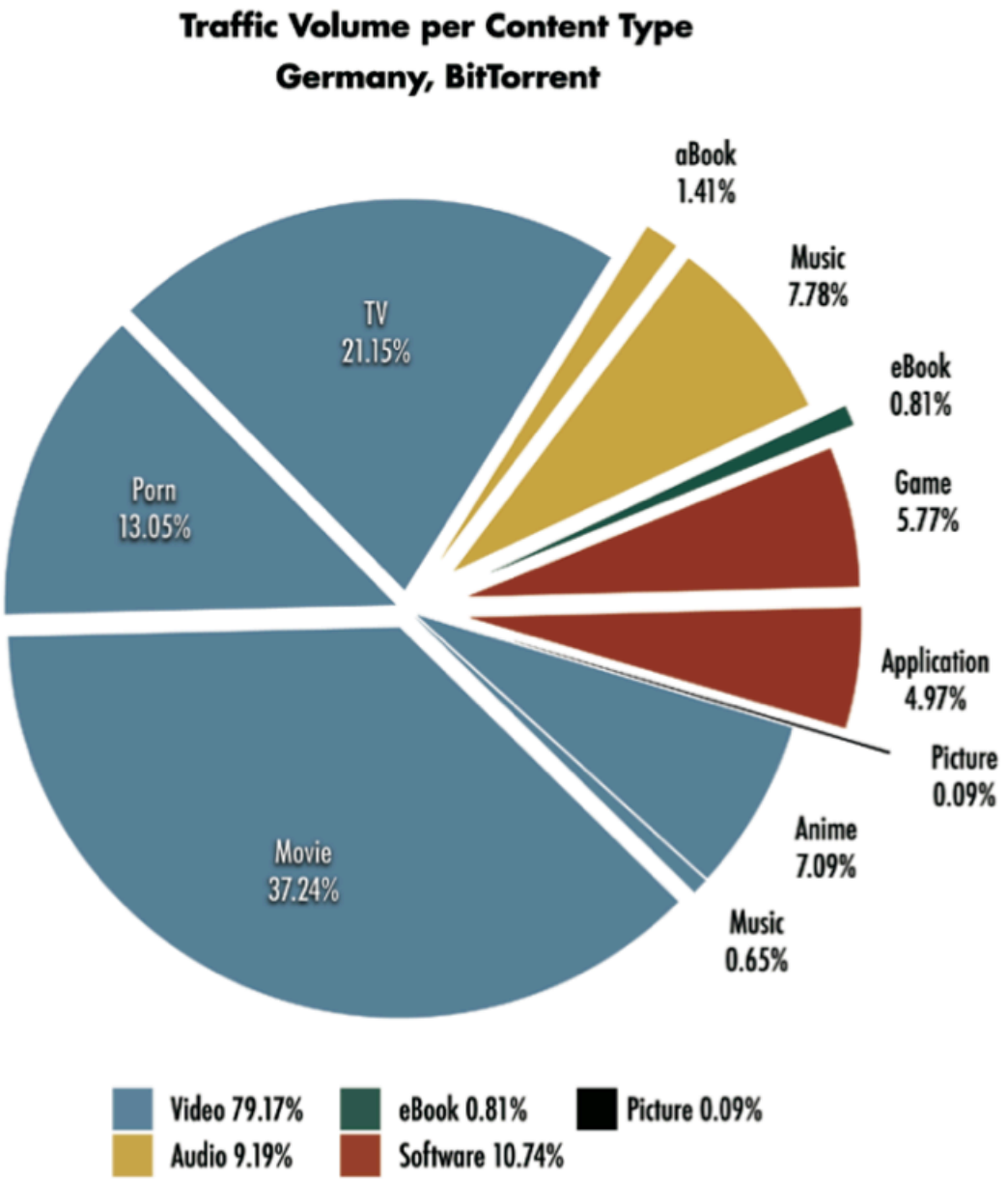
Source: Ipoque 2007

P2P Systems Germany 2007 by Volume



Source: Ipoque 2007

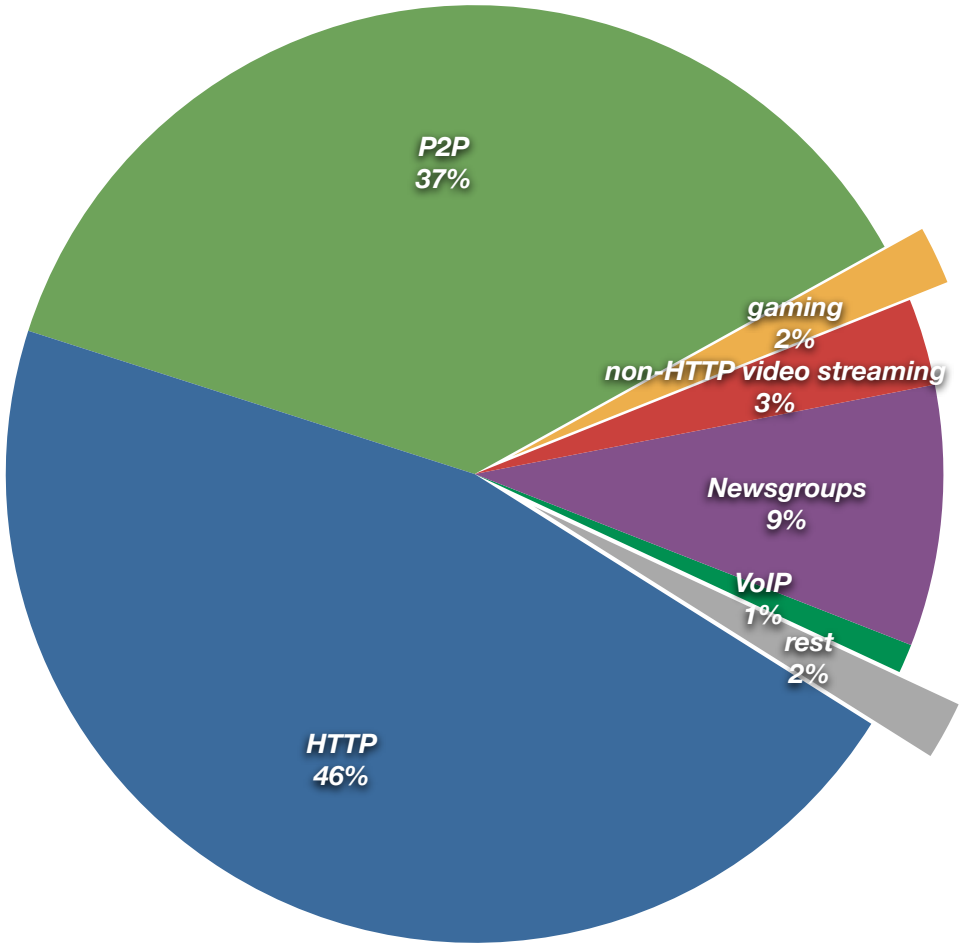
What Germans Download 2007 by Volume



Source: Ipoque 2007

Global Internet Traffic 2007

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



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

Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

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

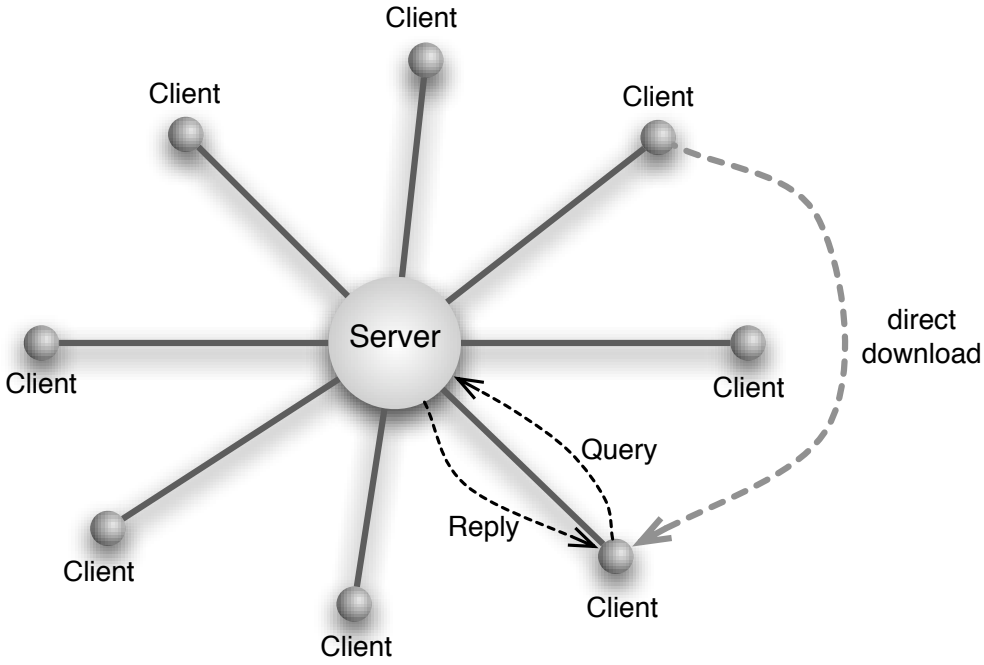
- **Distributed Hash-Tables (DHT) (1997)**
  - introduced for load balancing between web-servers
- **CAN (2001)**
  - efficient distributed DHT data structure for P2P networks
- **Chord (2001)**
  - efficient distributed P2P network with logarithmic search time
- **Pastry/Tapestry (2001)**
  - efficient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

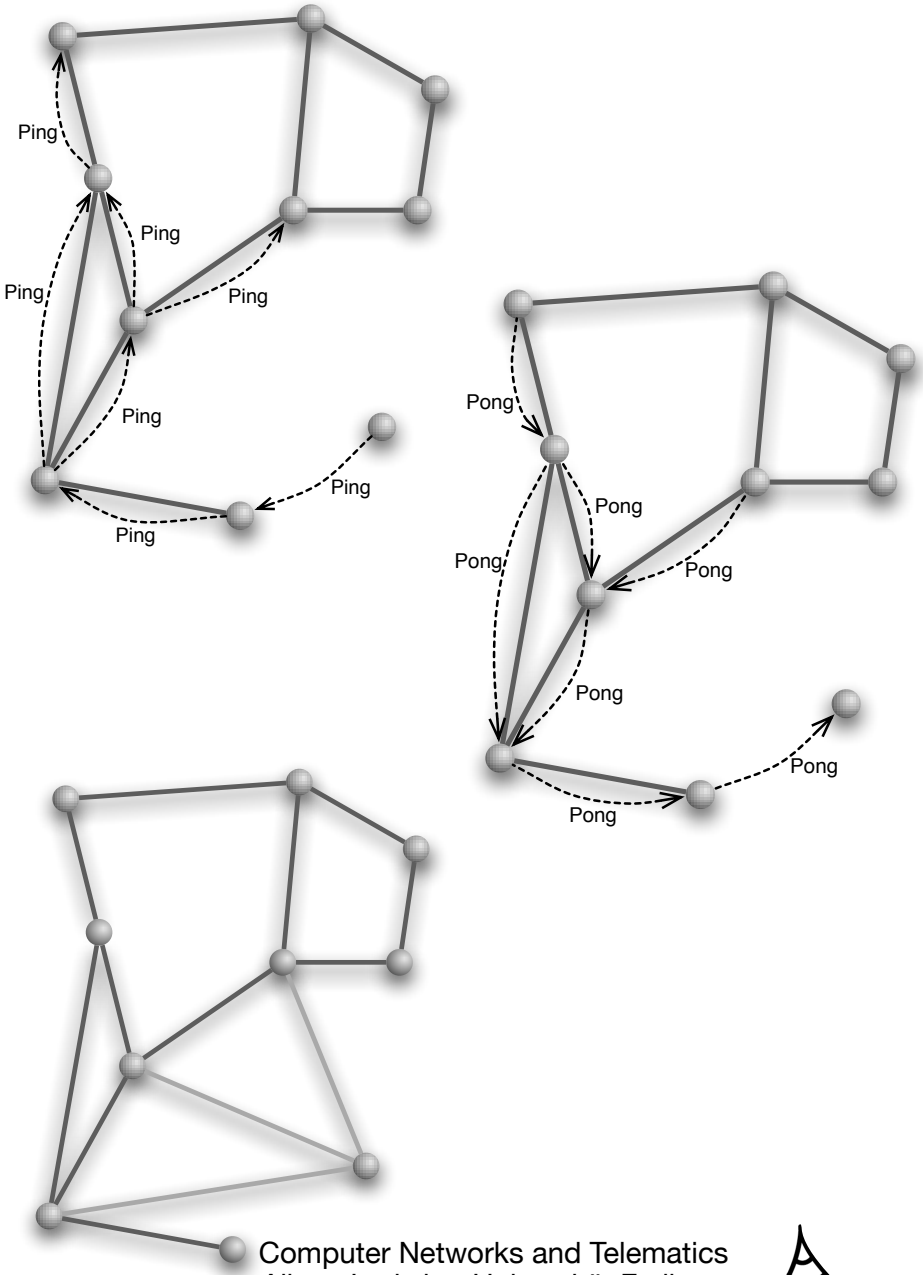


History of Gnutella

- **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

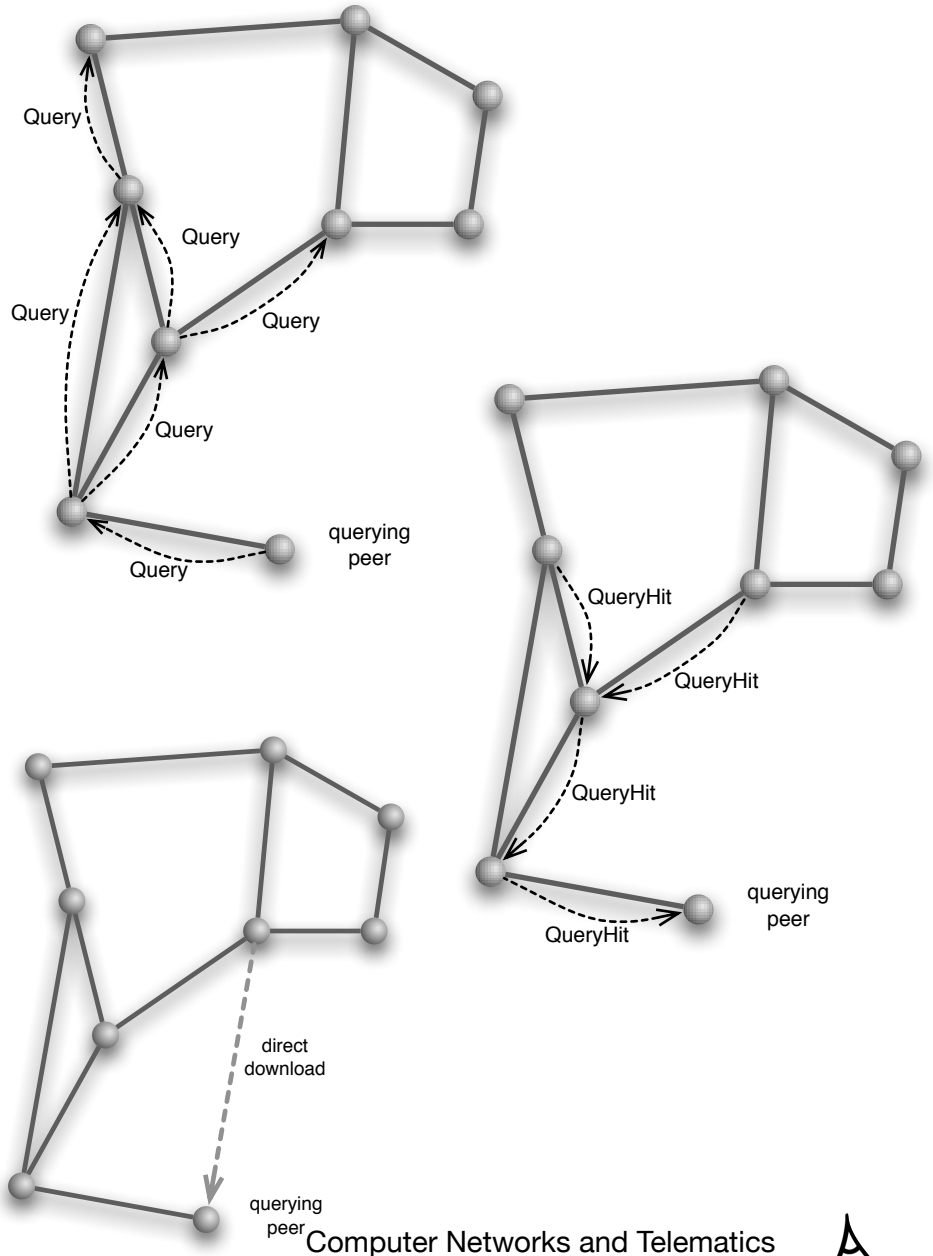
Gnutella — Connecting

- **Protokoll**
  - Ping
    - participants query for neighbors
    - are forwarded according for TTL steps (time to live)
  - Pong
    - answers Ping
    - is forwarded backward on the query path
    - reports IP and port adress (socket pair)
    - number and size of available files



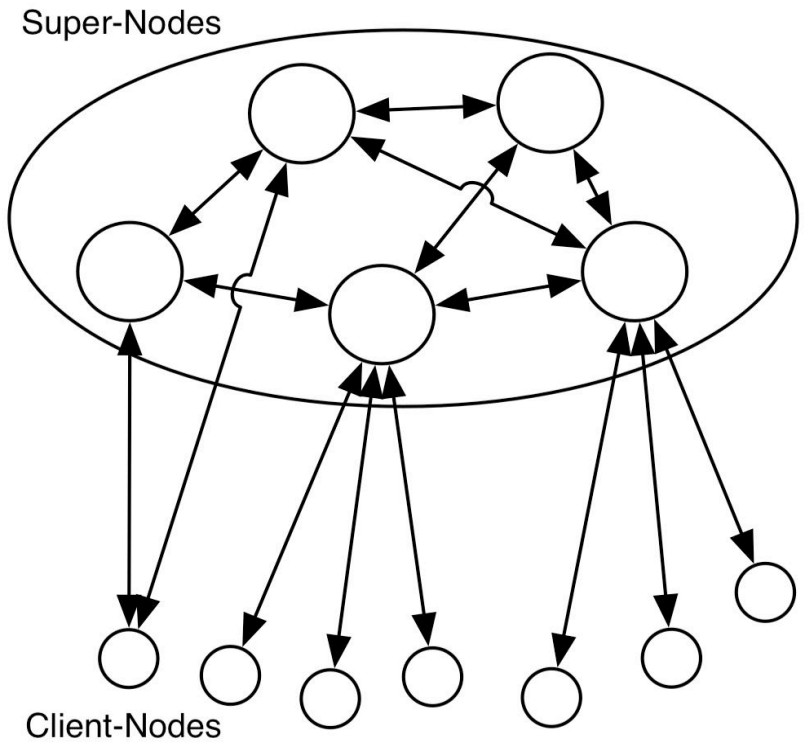
Gnutella — Query

- **File Query**
  - are sent to all neighbors
  - Neighbors forward to all neighbors
  - until the maximum hop distance has been reached
    - TTL-entry (time to live)
- **Protocol**
  - Query
    - for file for at most TTL hops
  - Query-hits
    - answers on the path backwards
- **If file has been found, then initiate direct download**



Gnutella 2

- **Hybride structure**
  - nodes with high bandwidth become P2P super nodes
  - super nodes provide P2P network like original Gnutella
  - normal nodes connect to the super nodes as clients
- **Used in**
  - FastTrack, Gnutella 2, Skype
- **Advantage**
  - improved scalability, lower latencies
- **Disadvantage**
  - clients can try to disable super node ability
  - scalability limited



FastTrack

- **Designed by Niklas Zennström, Janus Friies, Jaan Tallinn 2001**
  - authors of Skype (P2P-Internet-Telefonie)
- **Hybrid P2P Network**
  - Super nodes with special tasks
  - Software detects super node ability
    - e.g. no NAT server, more bandwidth, good network connection
  - super node for lookup
  - download by HTTP
    - from client to client (P2P)
- **Software**
  - not open source
  - official client contains malware
  - client super node communication investigated by reverse engineering
    - malware free clients available (Kazaa lite)
    - clients provide no super-node features
  - comunication between super nodes still unknown
- **Ares**
  - by the same developers
  - similar network
    - client: Ares Galaxy

E-Donkey

- **Client server index structure**
  - Server
    - special server software
      - e.g. Lugdunum, satan-edonkey-server
    - provide upload
    - store index information
  - Clients
    - several software clients
    - e.g. eMule, Shareaza, MLDonkey, eDonkey2000, Hydranode, Morpheus, ...
    - allow downloads from multiple other clients
    - some clients provide mechanisms for fair sharing
- **Discussion**
  - vulnerable for attacks
    - e.g. law enforcement, denial of service
    - in Feb 2006 Razorback2 server was confiscated by the Belgian police
  - Napster like P2P network
- **Sucessor Overnet**
  - was eliminated 2006
  - protocol still in use by a botnet

Napster Sucessors

- **OpenNap**
  - Napster clone
    - by reverse engineering of Napster communication
  - adds chatting tool
  - several clients available
  - star topology as in Napster
- **WinMX**
  - started as OpenNap client
  - adds hash code to Napster clone
  - in 2001 most successful P2P network in Japan
  - 2005 server farm was deactivated
    - and moved to Vanuatu
- **Soulseek**
  - Napster like system
  - adds features like
    - interest groups
    - chatting
    - wishing list
  - allows only single server

Direct Connect

- **Client-server network for indexing**
  - by NeoModus Inc.
  - server provide channels
  - channels can be secured by passwords
    - loophole against copyright laws
  - clients connect to server and transfer data directly from peer to peer
  - very popular in Scandinavia
- **Server is bottleneck**
- **Client software**
  - DC++, MLDonkey, NeoModus Direct Connect, ShakesPeer,...
- **Hub software**
  - Direct Connect Hub link, Hexhub, Open Direct Connect Hub, PtokaX,...

Skype

- **Usage**
  - VoIP, video, chat, file transfer, audio and video conferences
  - aims at a legal and free peer-to-peer communication platform
- **Hybrid Peer-to-Peer Network**
  - with super nodes
  - by the creators of Kazaa
- **Client-sever network**
  - for authentification and registration
- **Obscurity layers**
  - software encoded
  - communication encoded, possibly readable by Skype
- **Reverse engineering of Skype failed so far**
- **Little information by Skype about**
  - internal structure
  - privacy control
  - security
- **Popularity**
  - wide spread in private use
  - companies and organizations are reluctant to grant the usage of Skype
    - because of security concerns

Legal Situation

- **“IAAL*: What Peer-to-Peer Developers Need to Know about Copyright Law“, Fred von Lohmann, 2006**
- **Direct Infringement**
  - end users share files without authorization of the copyright owner
- **Secondary Infringement**
  - P2P tool maker
  - Inducement
    - if copyright infringement by third party is supported by the tool
    - with intent
  - Contributory infringement
    - knowingly contributes to another‘s infringement
  - Vicarious Liability
    - direct infringement by somebody
    - and right and ability to control by tool maker
    - and direct financial benefit
- **Defense strategies**
  - no direct infringement: „All users are innocent“
  - software capable of substantial non infringing uses
  - „safe harbors“ for online service providers
    - notifies users of a policy to terminate accounts of infringers
    - copyright agent for legal notices
    - must act after notice
    - must not know about infringements
    - must not receive direct benefits from infringements

Companies using P2P

- **Bittorrent, Kazaa**
- **Skype**
- **Microsoft, Intel, Sun**
- **Velocix, StratVantage, Loudcloud, Quiq, NextPage, Consilient, Sharman Networks, BlogAds, Octoshape, Joost, BigChampagne, Collanos, Vudu, ...**
- **ISPs and networking companies**
  - like Verizon, France Telecom, Pando, Norwegian Telecommunication ...
  - investigate possible usages of P2P networks

Companies & Organizations Fighting P2P Users

- **Fighting copyright regulations, e.g.**
  - Copyright Solutions
    - Blackwidow
    - P2P monitoring systems
    - monitors several systems
  - Evidenzia
    - monitors Bittorrent and eMule index files
- **RIAA (Recording Industry Association of America)**
- **IFPI (International Federation of Phonographic Industry)**
  - Digital File Check
    - program disables P2P Networks
  - legal actions against P2P servers and users
- **Law Attorneys**
  - bulk legal notices („Abmahnwelle“)
  - search file-sharing networks for specific contents and send legal notices threatening law suits
- **Internet Service Providers**
  - curb peer-to-peer traffic
  - e.g. Comcast & Bell, Canada
  - using deep packet inspection (DPI)
  - Anagram inc.
    - by Lawrence Roberts
    - P2P traffic detection (without DPI)

Problems of Copyright Enforcement

- **Automated detection and law enforcement**
  - Catherine Rampell, How It Does It: The RIAA Explains How It Catches Alleged Music Pirates, 2008
    - Blackwidow from Copyright Solutions
    - Evidenzia
  - Log in as fake user and collect IP addresses from other peers

- **Problem: False positives**
  - Michael Piatek, Tadayoshi Kohno, Arvind Krishnamurthy, Challenges and Directions for Monitoring P2P File Sharing Networks – or – Why My Printer Received a DMCA Takedown Notice
  - provoked RIAA accusation by peers blackmailing non-active („innocent“) IP addresses

Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

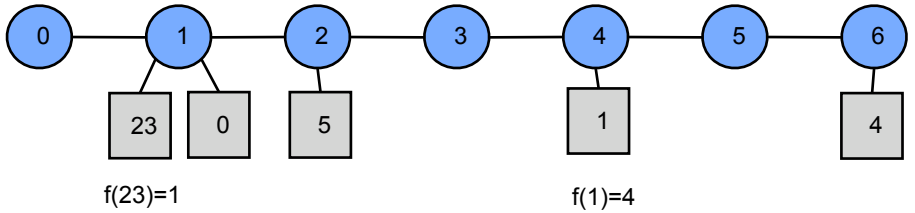
Distributed Hash Tables

- **DHT: The Method**
  - A data structure for P2P
- **Limitations of pure DHT**
  - Holes and Birthdays
- **Balancing DHTs**
  - Counting and Testing
- **Heterogeneous DHT**
  - Cones and Logarithms

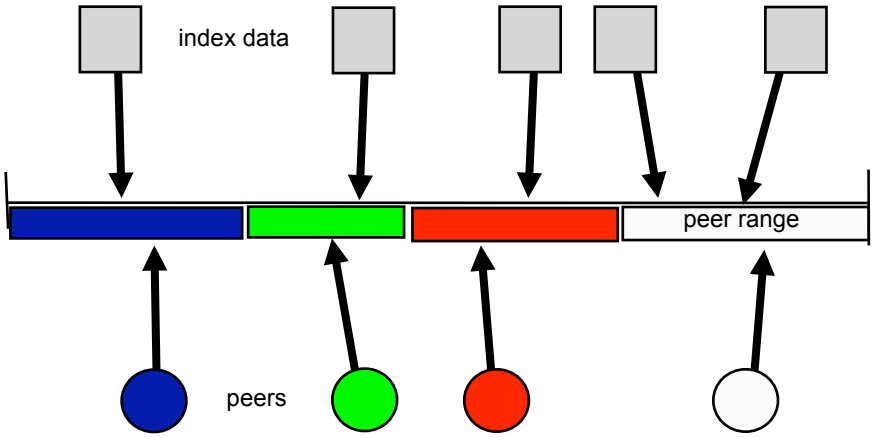
Distributed Hash-Table (DHT)

- **Hash table**
  - does not work efficiently for inserting and deleting
- **Distributed Hash-Table**
  - servers are „hashed“ to a position in an continuos set (e.g. line)
  - data is also „hashed“ to this set
- **Mapping of data to servers**
  - servers are given their own areas depending on the position of the direct neighbors
  - all data in this area is mapped to the corresponding server
- **Literature**
  - *“Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web”,* David Karger, Eric Lehman, Tom Leighton, Mathhew Levine, Daniel Lewin, Rina Panigrahy, STOC 1997

Pure (Poor) Hashing

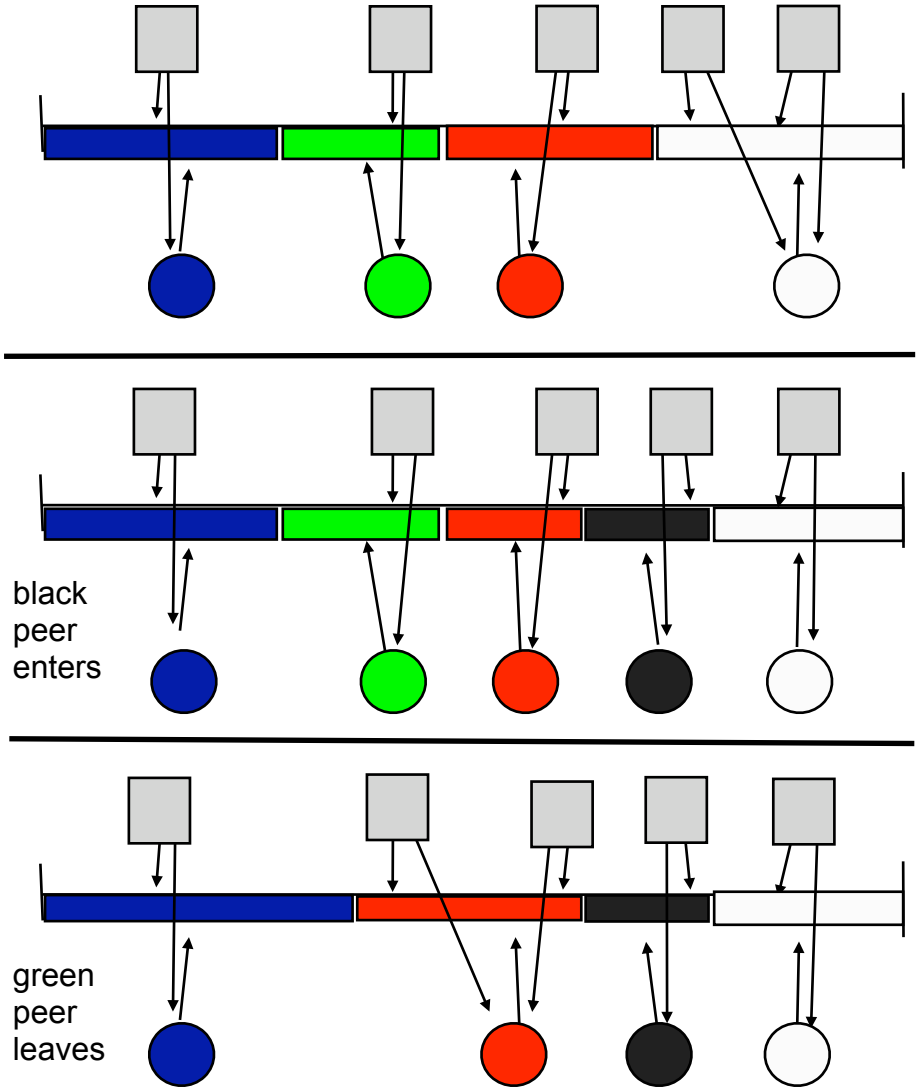


DHT

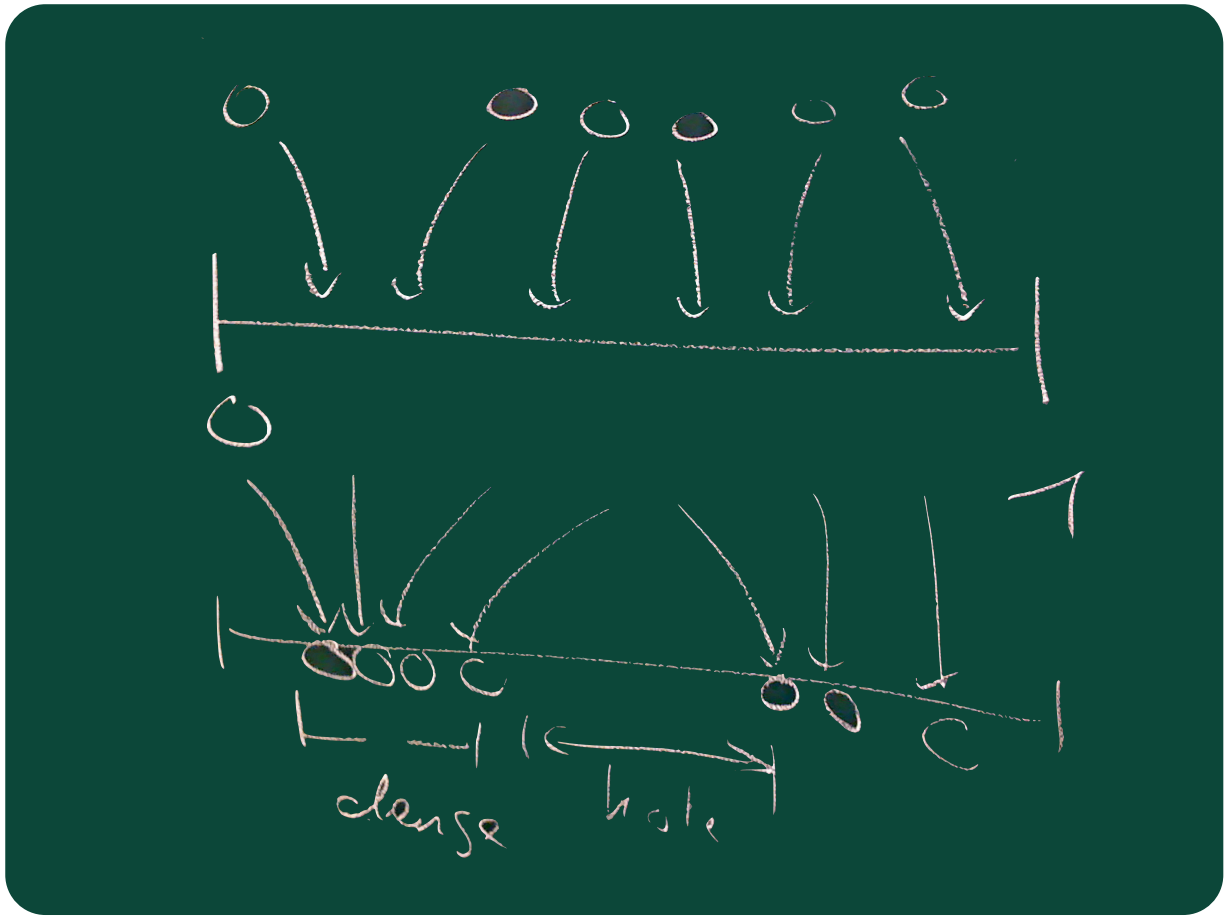


Entering and Leaving a DHT

- **Distributed Hash Table**
  - devices are hashed to to position
  - blocks are hashed according to the ID
- **When a device is added**
  - only blocks from neighbors have to be moved
- **When a device is deleted**
  - blocks are moved only to the neighbors



Holes and Dense Areas



Size of Holes

- **Theorem**
  - If n elements are randomly inserted into an array [0,1[then with constant probability there is a „hole“ of size Ω(log n/n), i.e. an interval without elements.
- **Proof**
  - Consider an interval of size log n / (4n)
  - The chance not to hit such an interval is (1-log n/(4n))
  - The chance that n elements do not hit this interval is

$$\left(1-\frac{\log n}{4n}\right)^n = \left(1-\frac{\log n}{4n}\right)^{\frac{4n}{\log n}\frac{\log n}{4}} \geq \left(\frac{1}{4}\right)^{\frac{1}{4}\log n} = \frac{1}{\sqrt{n}}$$

  - The expected number of such intervals is more than 1.
  - Hence the probability for such an interval is at least constant.

Proof of Dense Areas

$$\left(\frac{1}{4}\right)^{\frac{1}{4}\log n} = 2^{\left(\frac{1}{4}\log n\right)\log_2 \frac{1}{4}} \qquad \left(\log_2 \frac{1}{4} = -2\right)$$

$$= 2^{\left(-\frac{1}{2}\right)\cdot \log n}$$

$$= n^{-\frac{1}{2}}$$

$$= \frac{1}{\sqrt{n}}$$

Expectation: $\frac{4n}{\log n} \cdot \frac{1}{\sqrt{n}} = \frac{4\sqrt{n}}{\log n}$

Dense Spots

- **Theorem**
  - If n elements are randomly inserted into an array [0,1[then with constant probability there is a dense interval of length 1/n with at least Ω(log n/ (log log n)) elements.
- **Proof**
  - The probability to place exactly i elements in to such an interval is $\left(\frac{1}{n}\right)^i \left(1-\frac{1}{n}\right)^{n-i} \binom{n}{i}$
  - for i = c log n / (log log n) this probability is at least $1/n^k$ for an appropriately chosen c and k<1
  - Then the expected number of intervals is at least 1

Averaging Effect

- **Theorem**
  - If $\Theta(n \log n)$ elements are randomly inserted into an array [0,1[then with high probability in every interval of length 1/n there are $\Theta(\log n)$ elements.

Chernoff-Bound

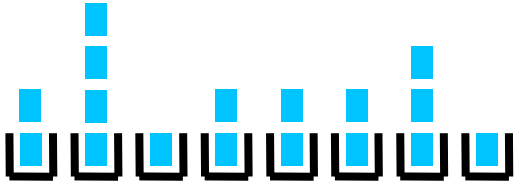
- **Theorem Chernoff Bound**
  - Let $x_1,...,x_n$ independent Bernoulli experiments with
    - $P[x_i = 1] = p$
    - $P[x_i = 0] = 1-p$
  - Let $S_n = \sum_{i=1}^{n} x_i$
  - Then for all c>0

$$\mathbf{P}[S_n \geq (1+c) \cdot \mathbf{E}[S_n]] \quad \leq \quad e^{-\frac{1}{3}\min\{c,c^2\}pn}$$

  - For 0≤c≤1

$$\mathbf{P}[S_n \leq (1-c) \cdot \mathbf{E}[S_n]] \quad \leq \quad e^{-\frac{1}{2}c^2pn}$$

Balls and Bins



Lemma

If m= k n ln n Balls are randomly placed in n bins:

1. Then for all c>k the probability that more than c ln n balls are in a bin is at most $O(n^{-c'})$ for a constant c'>0.
2. Then for all c<k the probability that less than c ln n balls are in a bin is at most $O(n^{-c'})$ for a constant c'>0.

Proof:

Consider a bin and the Bernoulli experiment B(k n ln n,1/n) and expectation: μ = m/n = k ln n

1. Case: c>2k

$$\begin{aligned}\mathrm{P}[X \geq c \ln n] &= \mathrm{P}[X \geq (1+(c/k-1))k \ln n] \\ &\leq e^{-\frac{1}{3}(c/k-1)k \ln n} \leq n^{-\frac{1}{3}(c-k)}\end{aligned}$$

2. Case: k<c<2k

$$\begin{aligned}\mathrm{P}[X \geq c \ln n] &= \mathrm{P}[X \geq (1+(c/k-1))k \ln n] \\ &\leq e^{-\frac{1}{3}(c/k-1)^2 k \ln n} \leq n^{-\frac{1}{3}(c-k)^2/}\end{aligned}$$

3. Case: c<k

$$\begin{aligned}\mathrm{P}[X \leq c \ln n] &= \mathrm{P}[X \leq (1-(1-c/k))k \ln n] \\ &\leq e^{-\frac{1}{2}(1-c/k)^2 k \ln n} \leq n^{-\frac{1}{2}(k-c)^2/k}\end{aligned}$$

Concept of High Probability

Lemma

If A(i) holds with **high** probability, i.e. $1-n^{-c}$, then

(A(1) and A(2) and ... and A(n)) with **high** probability,
i.e. $1-n^{-(c-1)}$

Proof:

- For all i: $P[\neg A(i)] \leq n^{-c}$
- Hence: $P[\neg A(1) \text{ or } \neg A(2) \text{ or } \ldots \neg A(n)] \leq n \cdot n^{-c}$

$$P[\neg(\neg A(1) \text{ or } \neg A(2) \text{ or } \ldots \neg A(n))] \leq 1 - n \cdot n^{-c}$$

DeMorgan:

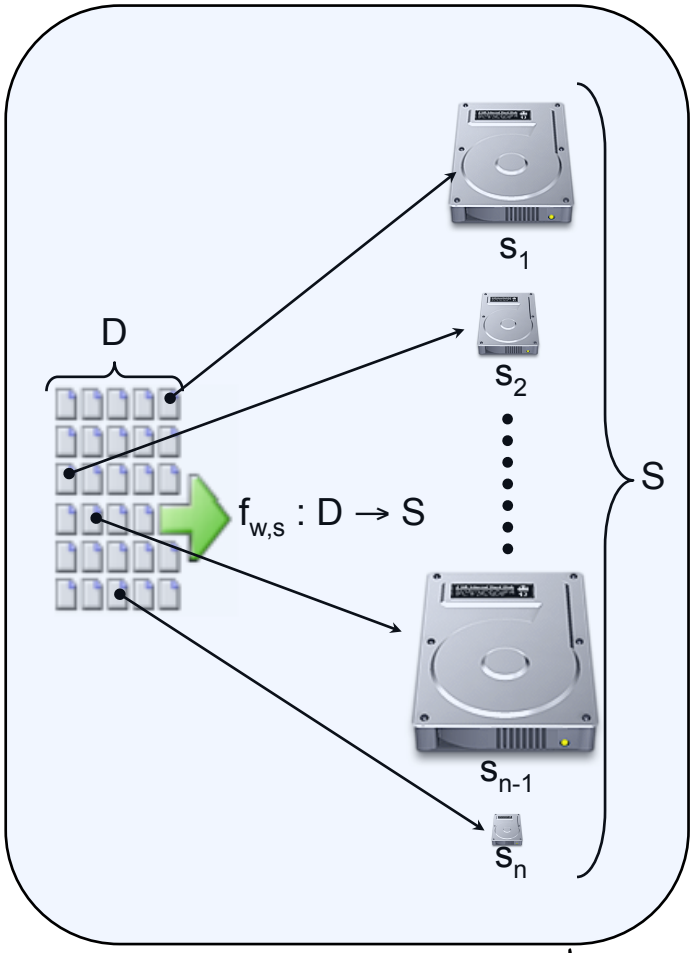
$$P[A(1) \text{ and } A(2) \text{ and } \ldots A(n)] \leq 1 - n \cdot n^{-c}$$

Principle of Multiple Choice

- **Before inserted check c log n positions**
- **For position p(j) check the distance a(j) between potential left and right neighbor**
- **Insert element at position p(j) in the middle between left and right neighbor, where a(j) was the maximum choice**
- **Lemma**
  - After inserting n elements with high probability only intervals of size 1/(2n), 1/n und 2/n occur.

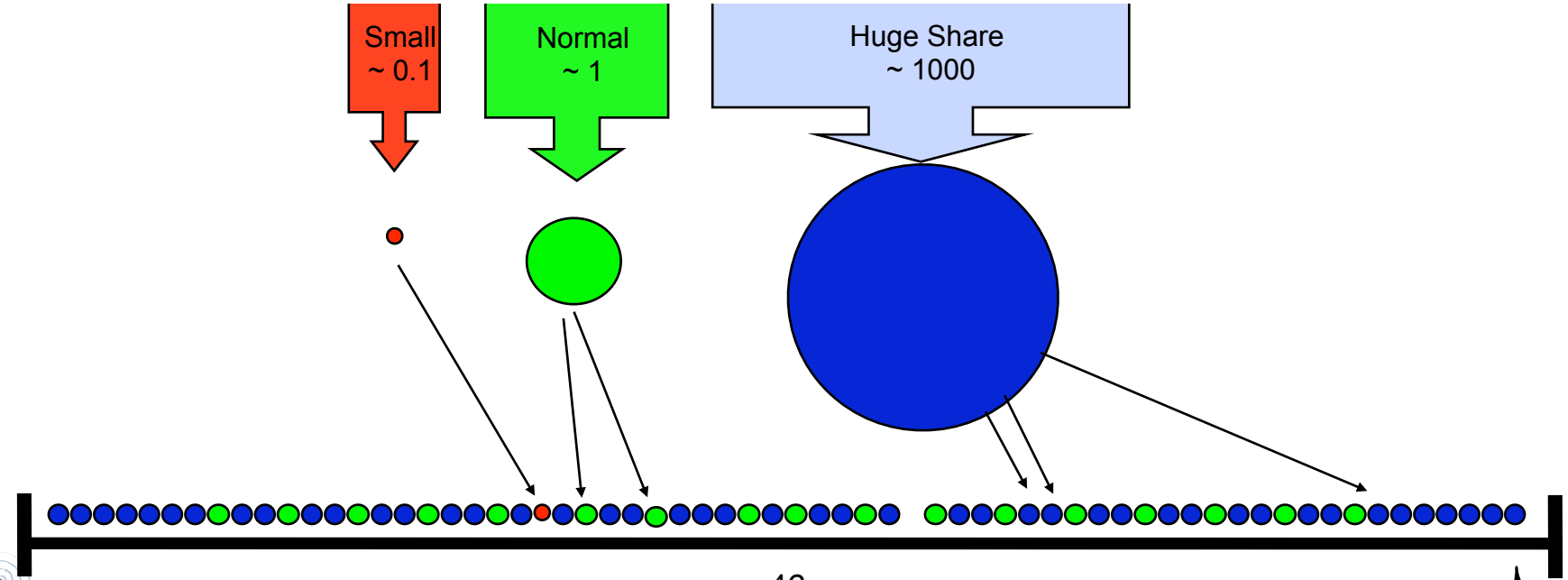
The Heterogeneous Case

- **Given**
  - a dynamic set of n nodes $V = \{v_1, \ldots, v_n\}$
  - dynamic weights $w : V \rightarrow \mathbf{R}^+$
  - dynamic set of data elements $X = \{x_1,\ldots,x_m\}$
- **Find a mapping $f_{w,V} : X \rightarrow V$**
- **With the following properties**
  - The mapping is **simple**
    - $f_{w,V}(x)$ be computed using V, x, w
    - without the knowledge of X\{x}
  - **Fairness**: for all $u,v \in V$:
    - $| f_{w,V}^{-1}(u)|/w(u) \approx | f_{w,V}^{-1}(v)|/w(v)$
  - **Consistency**:
    - minimal replacements to preserve the data distribution
- **where $f_{w,V}^{-1}(v) := \{ x \in X : f_{w,V}(x) = v \}$**



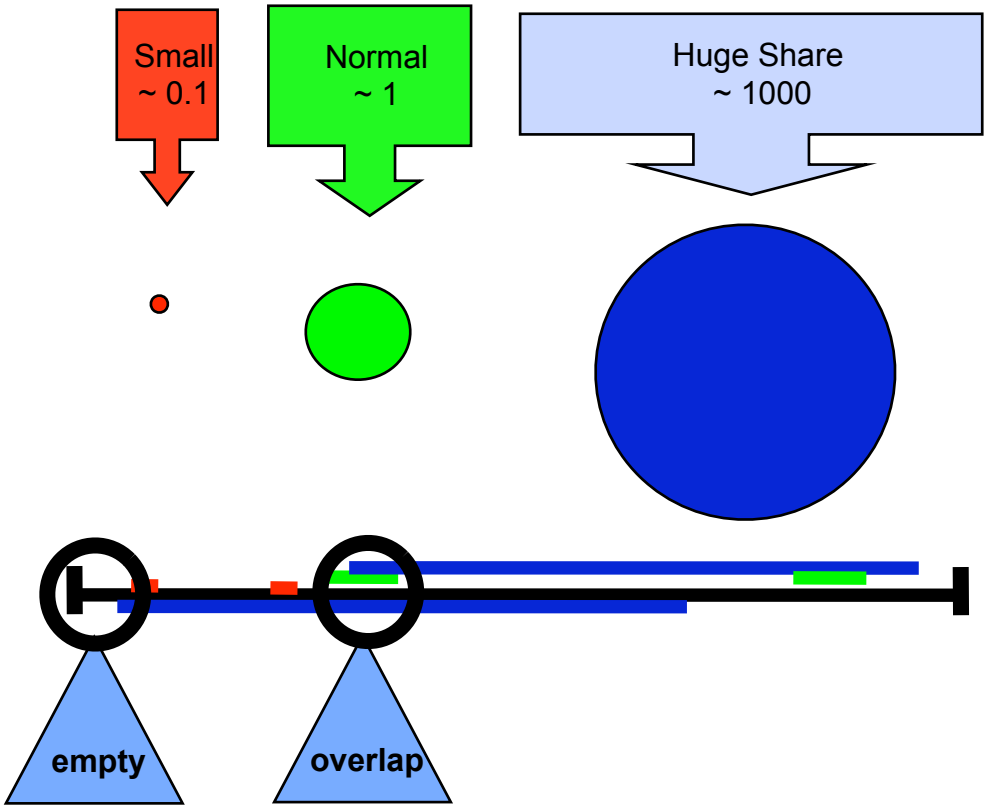
The Naive Approach to DHT

- Use $\left\lceil \frac{w_i}{\min_{j \in V}\{w_j\}} \right\rceil$ copies for each node w_i
- This is not feasible, if $\max_{j \in V}\{w_j\} / \min_{j \in V}\{w_j\}$ is too large
- Furthermore, inserting nodes with small weights increases the number of copies of all nodes.



SIEVE: Interval based consistent hashing

- **Interval based approach**
  - André Brinkmann, Kay Salzwedel, Christian Scheideler, Compact, Adaptive Placement Schemes for Non-Uniform Capacities, 14th ACM Symposium on Parallelism in Algorithms and Architectures 2002 (SPAA 2002)
- **Map nodes to random intervals (via hash function)**
  - interval length proportional to weight
- **Map data items to random positions (via hash function)**
- **Two problems**
  - What to do if intervals overlap?
  - What to do if the unions of intervals do not overlap the hash space M?

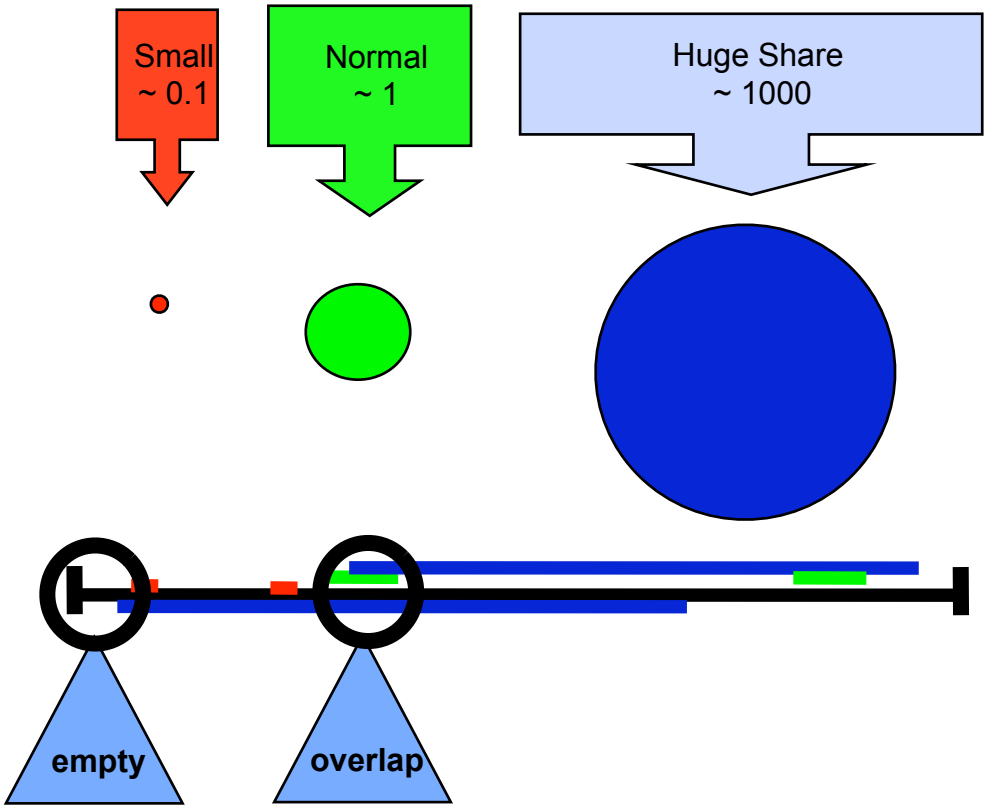


SIEVE: Interval based consistent hashing

1. **What to do if intervals overlap?**
   - Uniformly choose random candidate from the overlapping intervals
2. **What to do if the unions of intervals do not overlap the hash space M?**
   - Increase all intervals by a constant factor (stretch factor)
   - Use O(log n) copies of all nodes
     - resulting in O(n log n) intervals

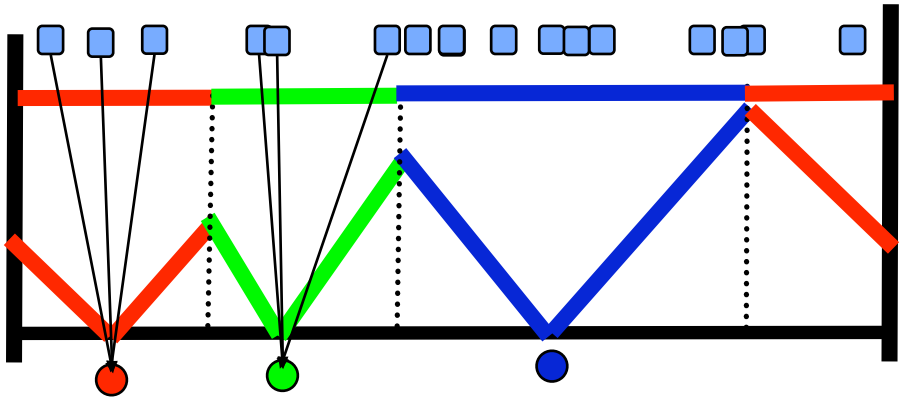
➢ **If more nodes appear**
   - then decrease all intervals by a constant factor

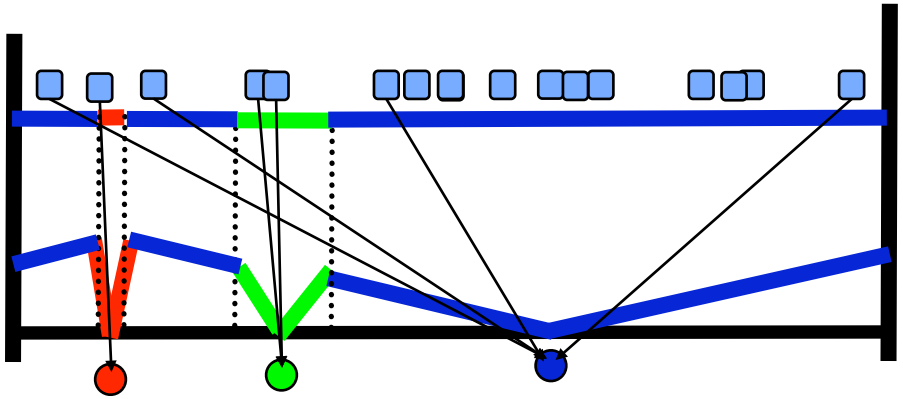
➢ **SIEVE is not providing monotony**
   - Re-stretching leads to unnecessary re-assignments



The Linear Method

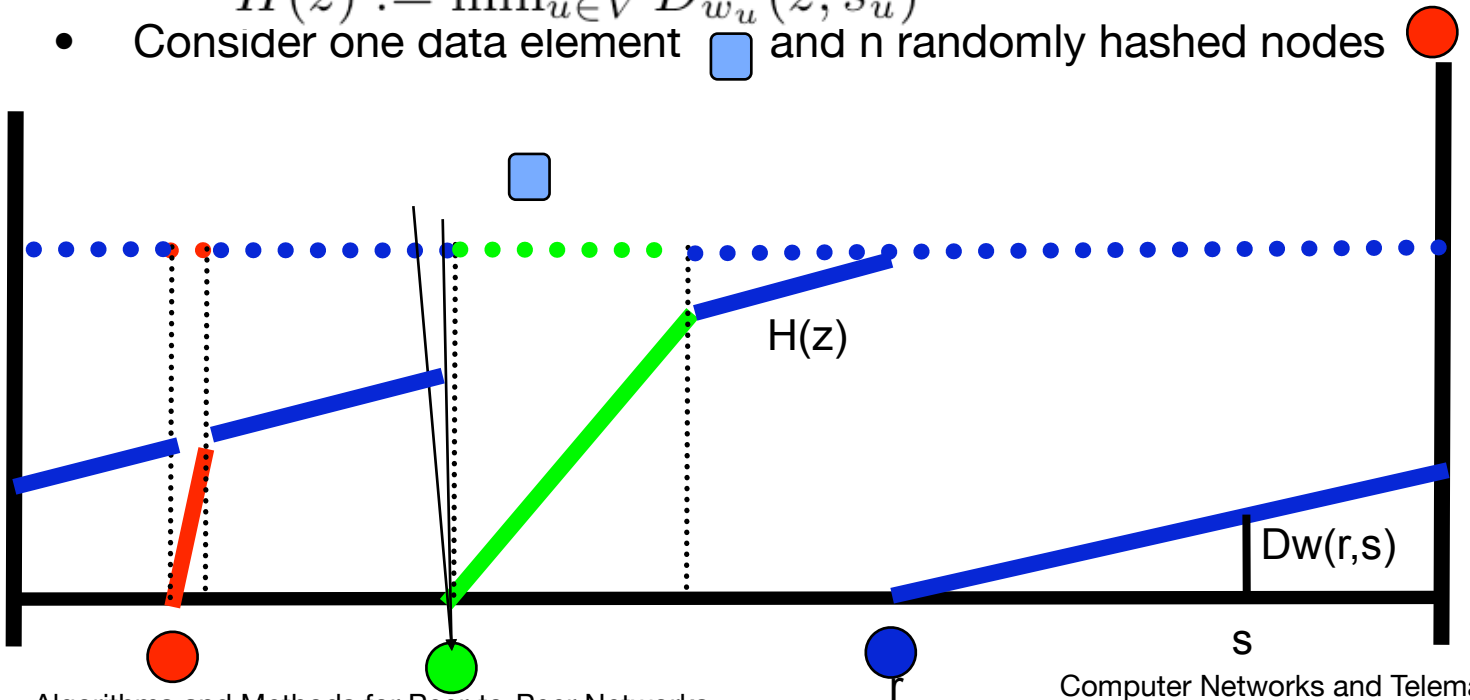
- Christian Schindelhauer, Gunnar Schomaker, Weighted Distributed Hash Tables, 17th ACM Symposium on Parallelism in Algorithms and Architectures 2005 (SPAA 2005)
- **Alternative presentation of (uniform) Consistent Hashing**
- **After "randomly" placing nodes into M**
  - Add cones pointing to the node's location in M
- **Compute for each data element x the height of the cones**
  - Choose the cone with smallest height
- **For the Linear Method**
  - Choose for each node i a cone stretched by the factor wi
- **Compute for each data element x the height of the cones**
  - Choose the cone with smallest height





The Linear Method: Basics

- **For easier description we use half-cones,**
  - the weighted distance is $D_w(r,s) := \frac{((s-r) \bmod 1)}{w}$
    - where x mod 1 := x - ⌊x⌋
- **Analyzing heights is easier as analyzing interval lengths!**
- **Define:** $H(z) := \min_{u \in V} D_{w_u}(z, s_u)$
  - Consider one data element and n randomly hashed nodes

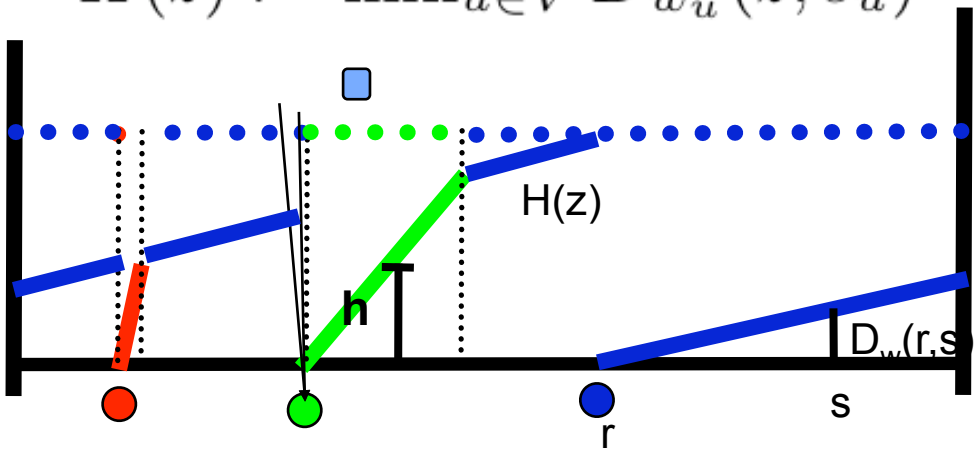


The Linear Method: Basics

LEMMA 1. *Given n nodes with weights $w_1, \ldots, w_n$. Then the height $H(r)$ assigned to a position r in M is distributed as follows:*

$$P[H(r) > h] = \begin{cases} \prod_{i \in [n]} (1 - h w_i) , & \text{if } h \leq \min_i \{\frac{1}{w_i}\} \\ 0 , & \text{else} \end{cases}$$

$$H(z) := \min_{u \in V} D_{w_u}(z, s_u)$$



The Linear Method with Copies

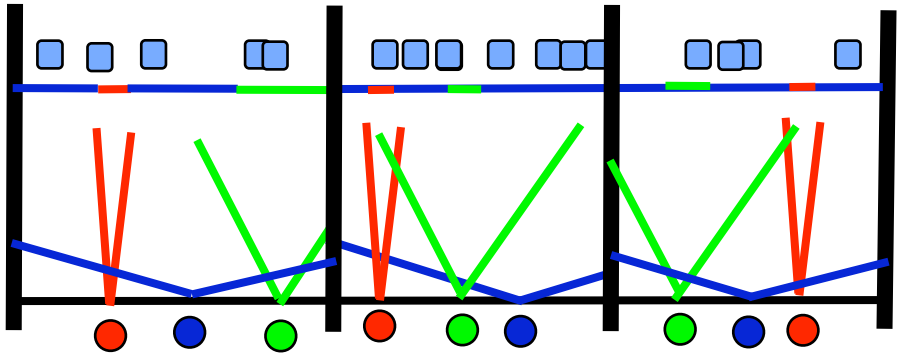
THEOREM 2. *Let $\epsilon > 0$. Then, the Linear Method using $\lceil \frac{2}{\epsilon} + 1 \rceil$ copies assigns one data element to node i with probability p_i where*

$$(1 - \sqrt{\epsilon}) \cdot \frac{w_i}{W} \leq p_i \leq (1 + \epsilon) \cdot \frac{w_i}{W} .$$

- **A constant number of copies suffice to "repair" the linear function**
- **This theorem works only for one data item**
  - If many data items are inserted, then the original bias towards some nodes is reproduced:
    - "Lucky" nodes receive more data items
- **Solution**
  - Independently repeat the game at least O(log n) times

Partitioning and the Linear Method

- **Partitions:**
  - Partition the hash range into sub-intervals
  - Map each data element into the whole interval
  - Map for each node 2/ε+1 copies into each sub-interval



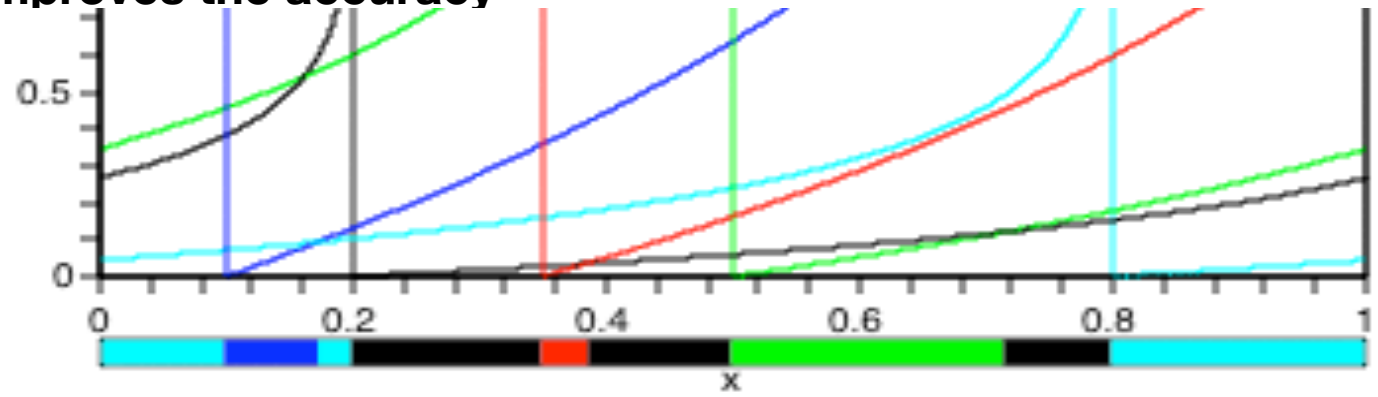
Theorem 3 *For all $\epsilon, \epsilon' > 0$ and $c > 0$ there exists $c' > 0$ such that when we apply the Linear Method to n nodes using $\lceil \frac{2}{\epsilon} + 1 \rceil$ copies and $c' \log n$ partitions, the following holds with high probability, i.e. $1 - n^{-c}$.*

Every node $i \in V$ receives all data elements with probability p_i such that

$$(1 - \sqrt{\epsilon} - \epsilon') \cdot \frac{w_i}{W} \leq p_i \leq (1 + \epsilon + \epsilon') \cdot \frac{w_i}{W} .$$

The Logarithmic Method

- **Replacing the linear function by** $L_w(r,s) := \frac{-\ln((1-(r-s)) \bmod 1)}{w}$
- **improves the accuracy**



FACT 2. *If in the Logarithmic Method (without copies and without partitions) a node arrives with weight w then the probability that data element x with previous height H_x is assigned to the new node is* $1 - e^{-wH_x}$.

THEOREM 6. Given n nodes with positive weights $w_1, \ldots, w_n$ the Logarithmic Method assigns a data element to node i with probability $\frac{w_i}{W}$, where $W := \sum_{i=1}^{|V|} w_i$.

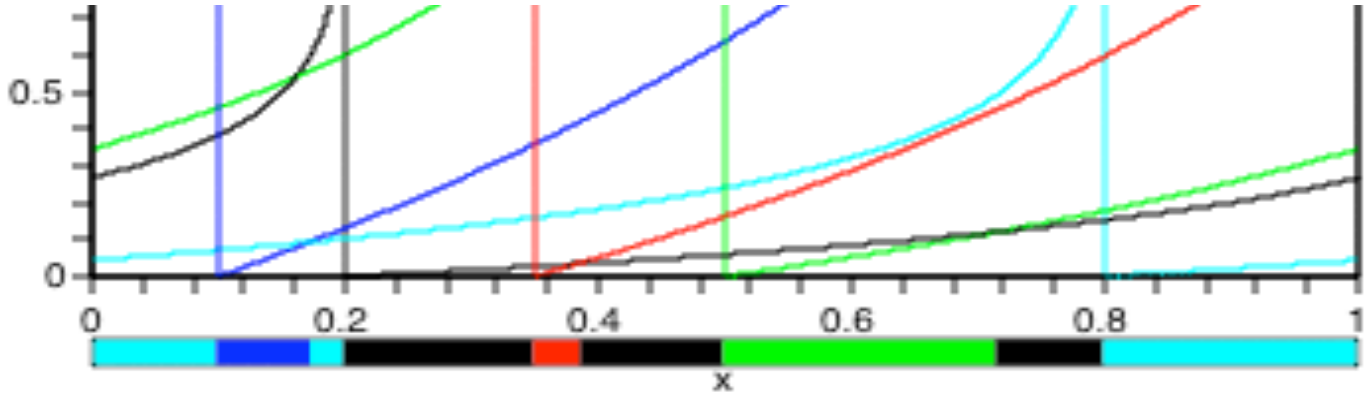
The Logarithmic Method

- **Replacing the linear function with -ln((1-$d_i(x)$) mod 1)/w_i improves the accuracy of the probability distribution**

Theorem 7 *For all $\epsilon > 0$ and $c > 0$ there exists $c' > 0$, where we apply the Logarithmic Method with $c' \log n$ partitions. Then, the following holds with high probability, i.e. $1 - n^{-c}$.*

Every node $i \in V$ receives data elements with probability p_i such that

$$(1-\epsilon) \cdot \frac{w_i}{W} \leq p_i \leq (1+\epsilon) \cdot \frac{w_i}{W} .$$



Further Features

- **Efficient data structure for the linear and logarithmic method**
  - can be implemented within O(n) space
  - Assigning elements can be done in O(log n) expected time
  - Inserting/deleting new nodes can be done in amortized time O(1)
- **Predicting Migration**
  - The height of a data element correlates with the probability that this data element is the next to migrate to a different server
- **Fading in and out**
  - Since the consistency works also for the weights:
  - Nodes can be inserted by slowly increasing the weight
  - No additional overhead
  - Node weight represents the transient download state
  - Vice versa for leaving nodes

Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Structured Networks

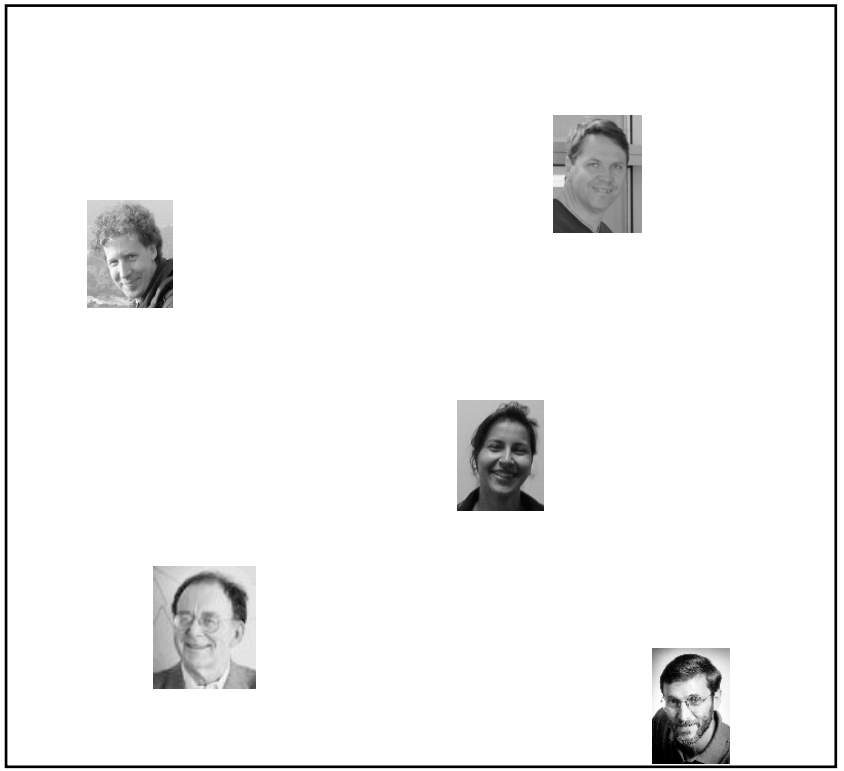
- **Standards**
  - CAN
  - Chord
  - Pastry
  - Tapestry
- **Extreme solutions**
  - Koorde
  - Distance Halving
  - Kelips

Peer-to-Peer Networks

Content Addressable Network (CAN)

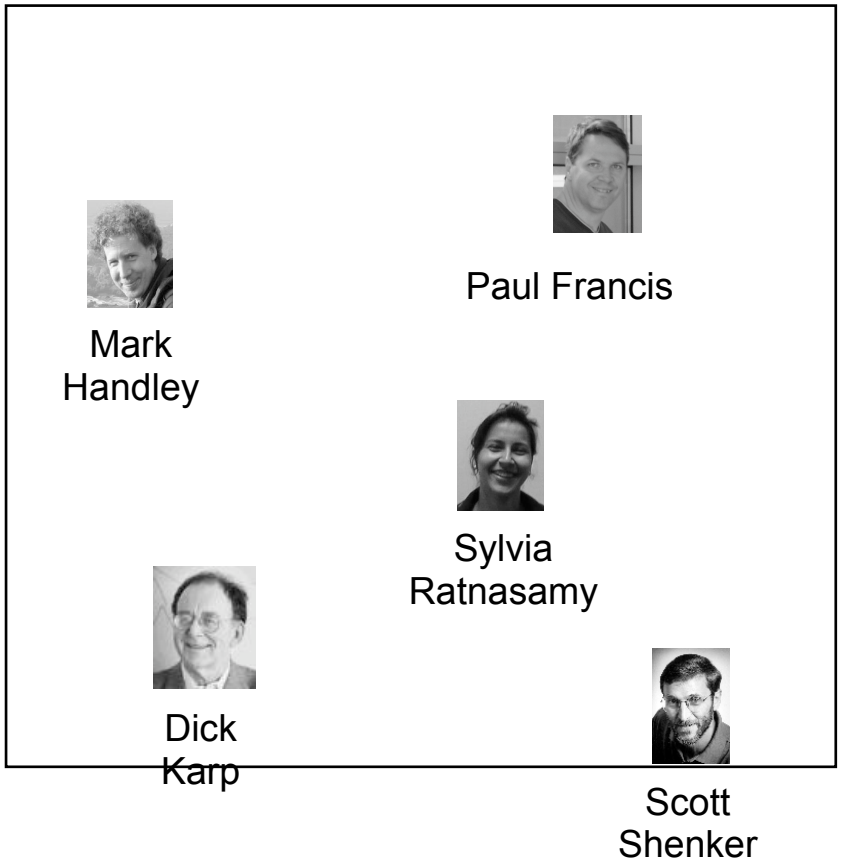
CAN Playground

- **Index entries are mapped to the square $[0,1]^2$**
  - using two hash functions to the real numbers
  - according to the search key
- **Assumption:**
  - hash functions behave a like a random mapping



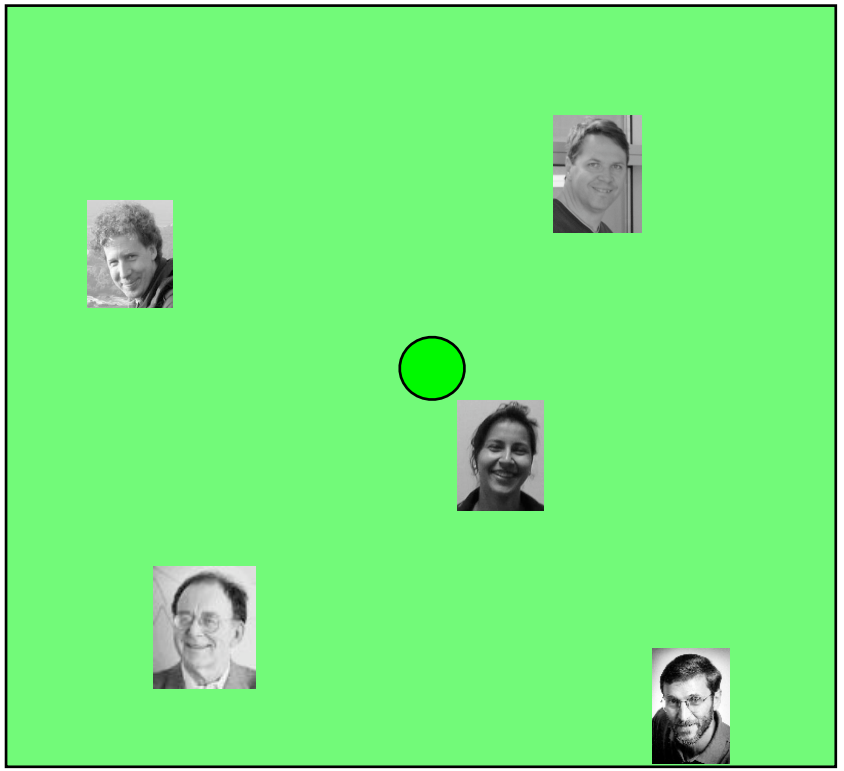
CAN Index Entries

- **Index entries are mapped to the square $[0,1]^2$**
  - using two hash functions to the real numbers
  - according to the search key
- **Assumption:**
  - hash functions behave a like a random mapping
- **Literature**
  - Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Computer Communication Review. Volume 31., Dept. of Elec. Eng. and Comp. Sci., University of California, Berkeley (2001) 161–172



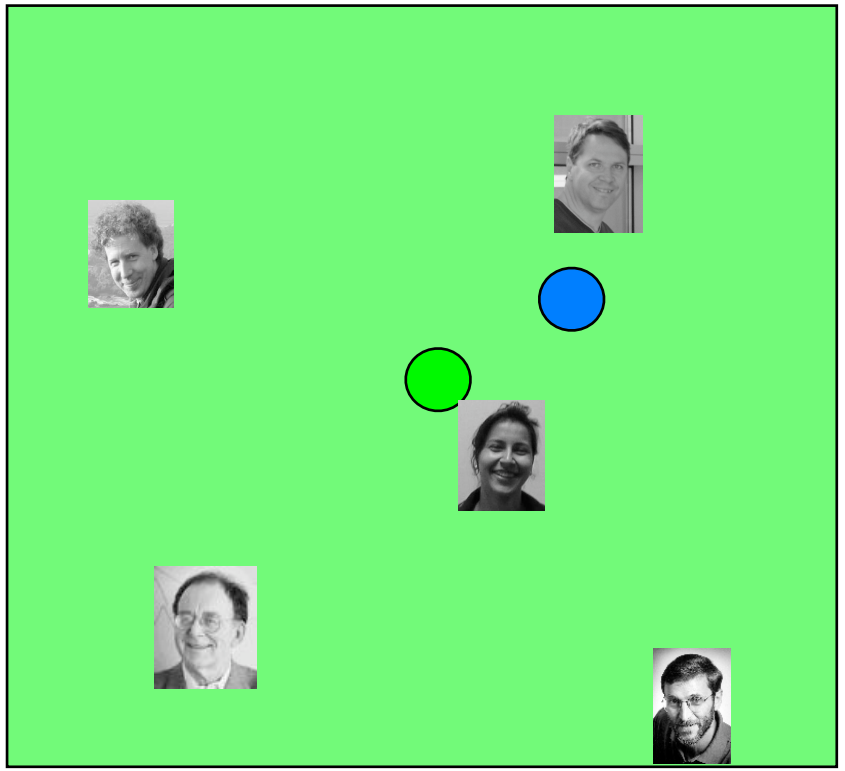
First Peer in CAN

- In the beginning there is one peer owning the whole square
- All data is assigned to the (green) peer



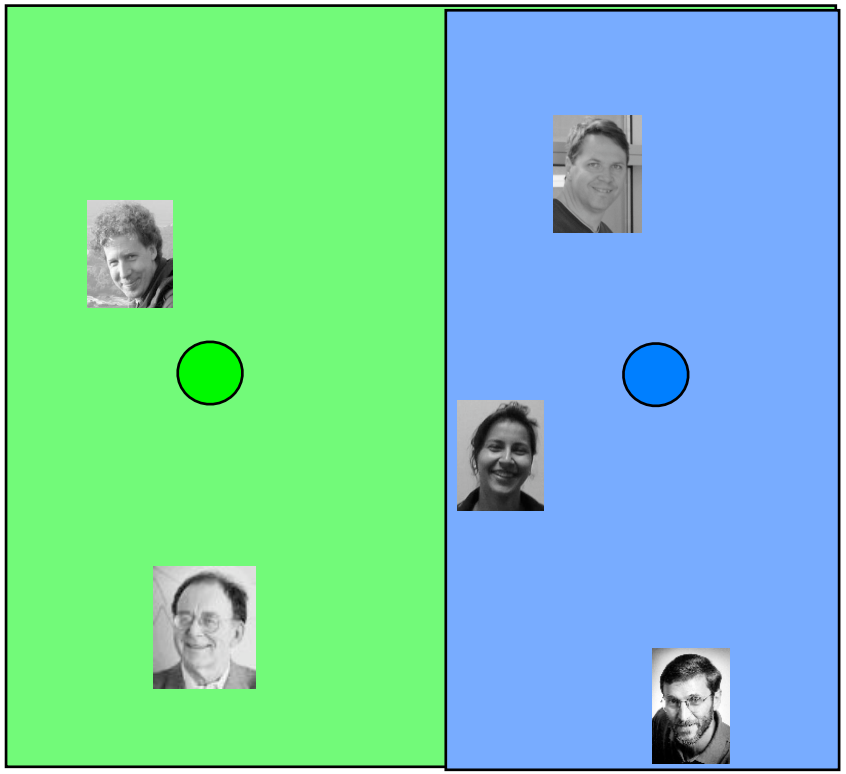
CAN: The 2nd Peer Arrives

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner



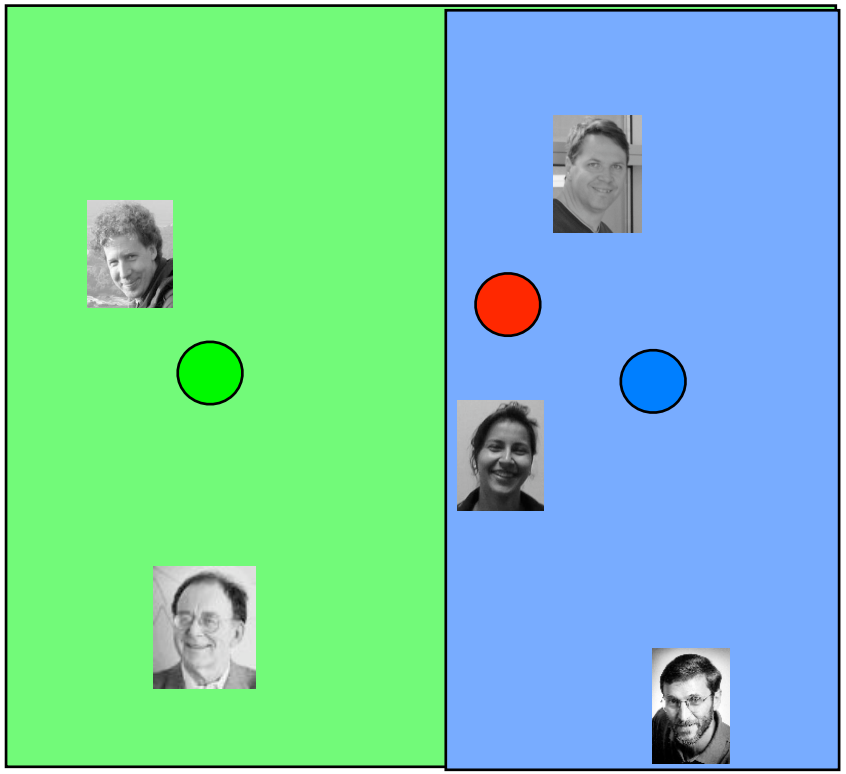
CAN: 2nd Peer Has Settled Down

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



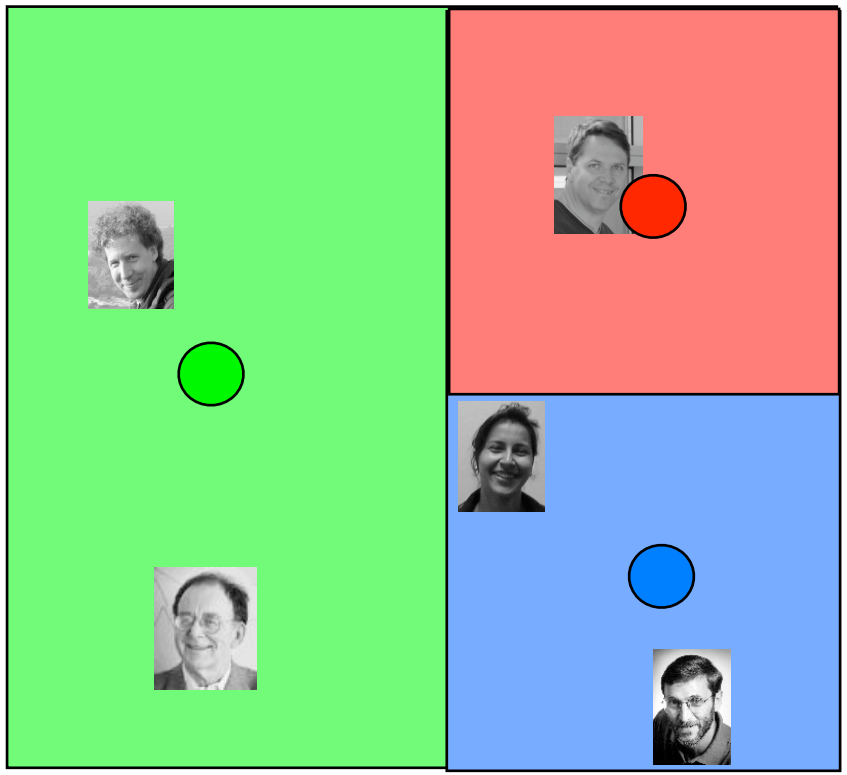
3rd Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



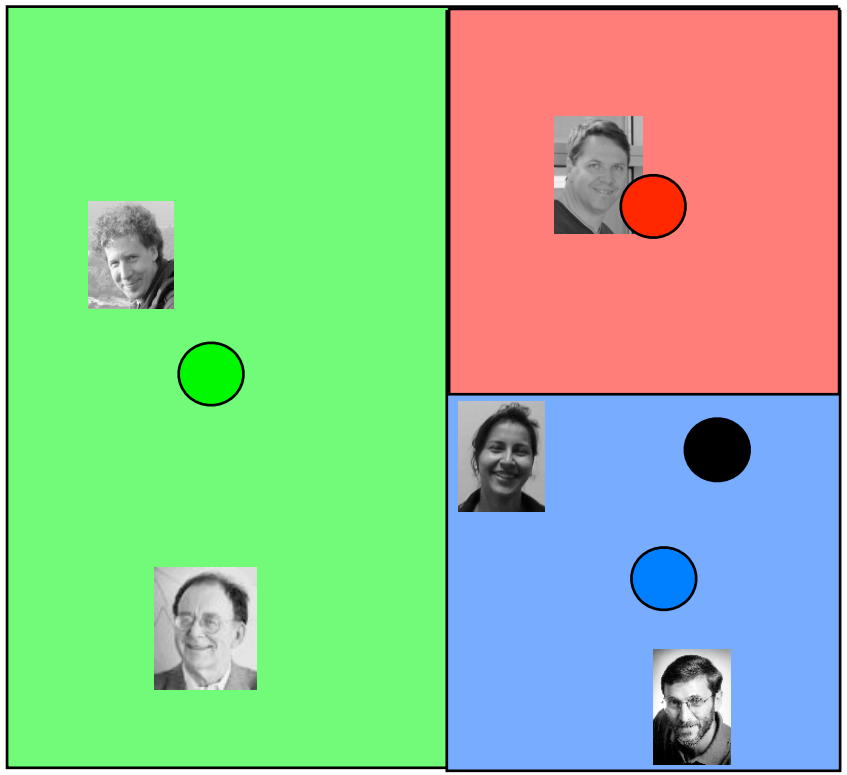
CAN: 3rd Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



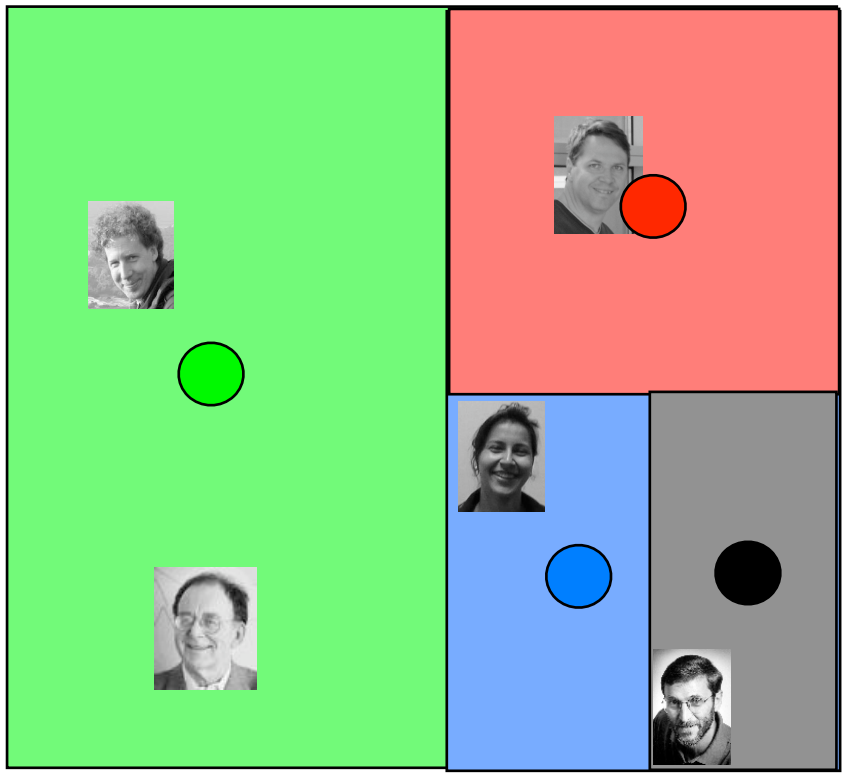
CAN: 4th Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



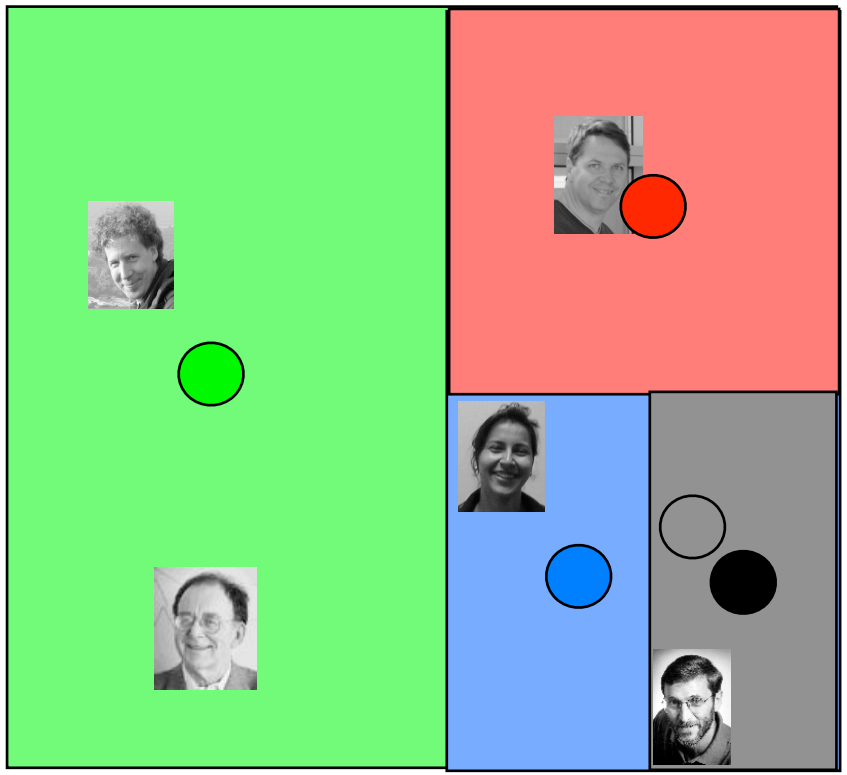
CAN: 4th Peer Added

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



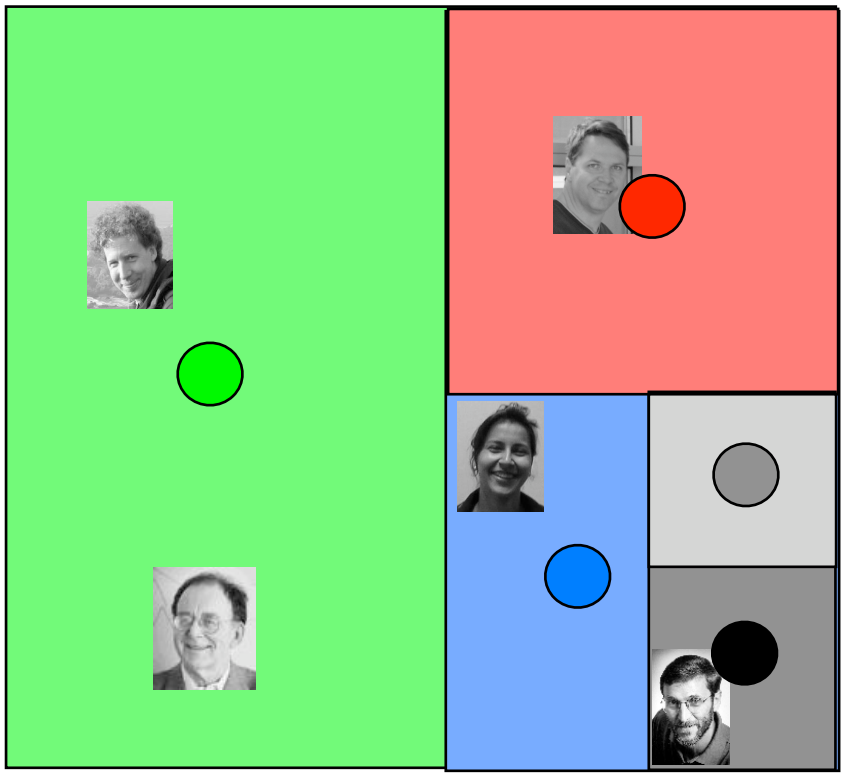
CAN: 5th Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



CAN: All Peers Added

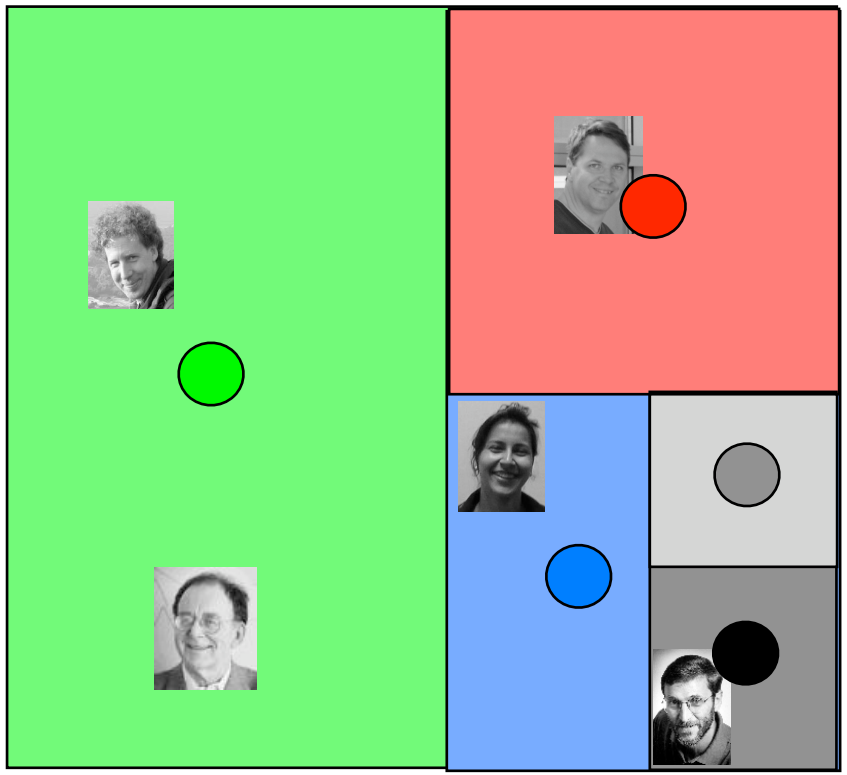
- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



On the Size of a Peer‘s Area

- **R(p): rectangle of peer p**
- **A(p): area of the R(p)**
- **n: number of peers**
- **area of playground square: 1**
- **Lemma**
  - For all peers we have $E[A(p)] = \frac{1}{n}$
- **Lemma**
  - Let $P_{R,n}$ denote the probability that no peers falls into an area R. Then we have

$$P_{R,n} \leq e^{-n\mathrm{Vol}(R)}$$

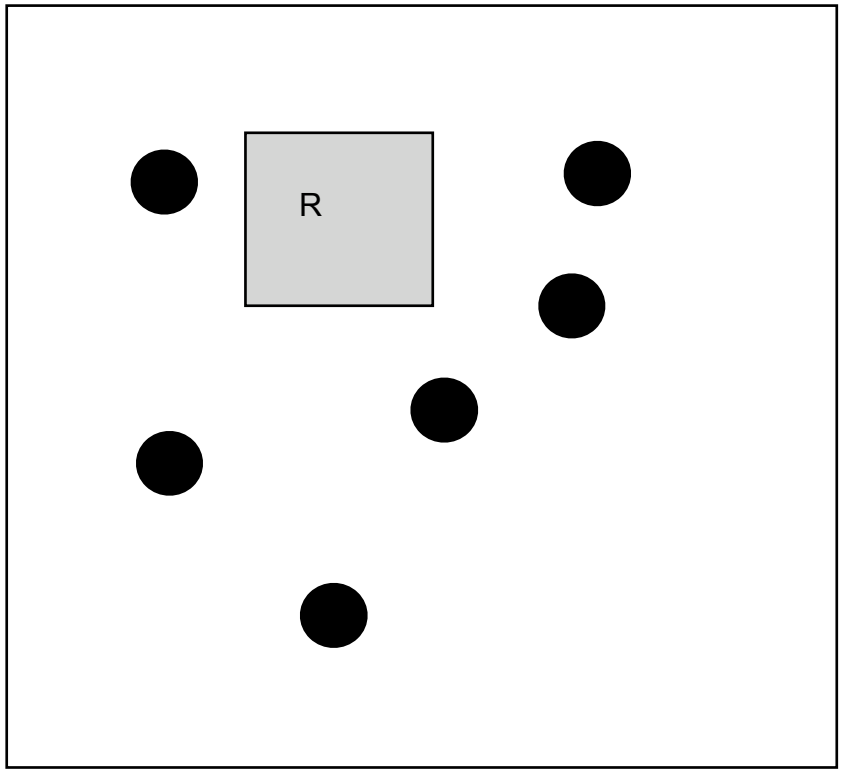


An Area Not being Hit

- **Lemma**
  - Let $P_{R,n}$ denote the probability that no peers falls into an area R. Then we have $P_{R,n} \leq e^{-n\mathrm{Vol}(R)}$
- **Proof**
  - Let x=Vol(R)
  - The probability that a peer does not fall into R is $1-x$
  - The probability that n peers do not fall into R is $(1-x)^n$
  - So, the probability is bounded by

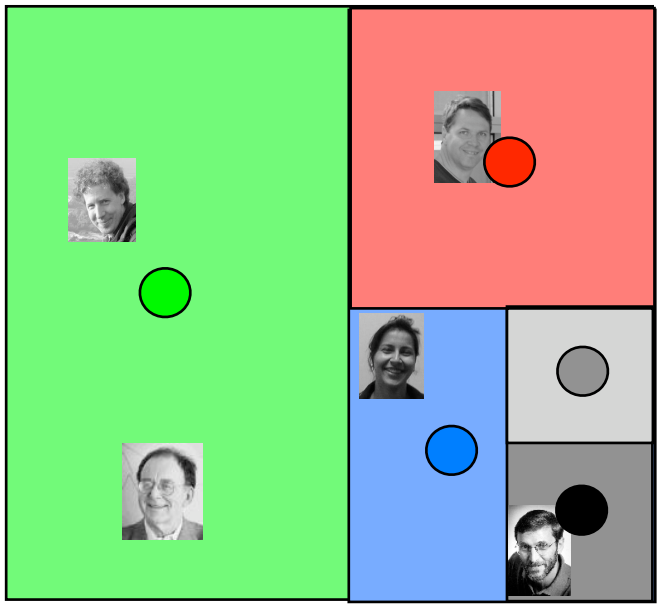
    $$(1-x)^n = ((1-x)^{\frac{1}{x}})^{nx} \leq e^{-nx}$$
  - because

    $$m > 1 \ : \ \left(1-\frac{1}{m}\right)^m \leq \frac{1}{e}$$



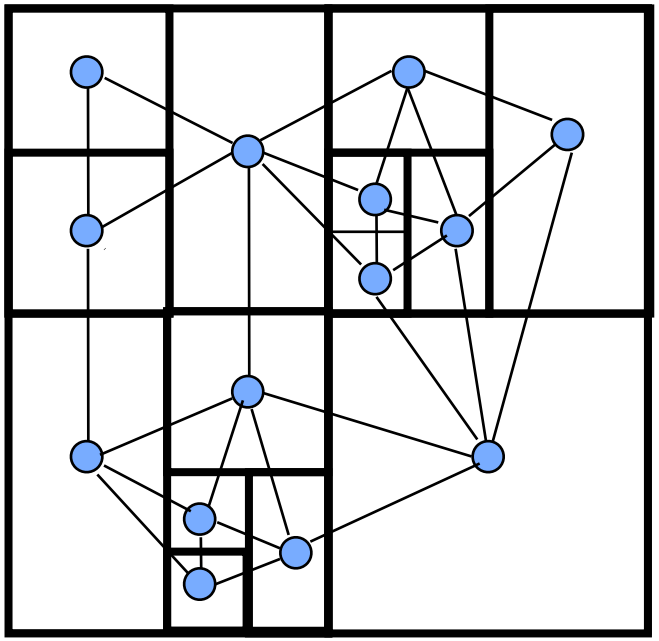
How Fair Are the Data Balanced

- **Lemma**
  - With probability n^{-c} a rectangle of size (c ln n)/n is not further divided
- **Proof**
  - Let $P_{R,n}$ denote the probability that no peers falls into an area R. Then we have $P_{R,n} \leq e^{-n\mathrm{Vol}(R)}$
- **Every peer receives at most c (ln n) m/n elements**
  - if all m elements are stored equally distributed over the area
- **While the average peer stores m/n elements**
- **So, the number of data stored on a peer is bounded by c (ln n) times the average amount**



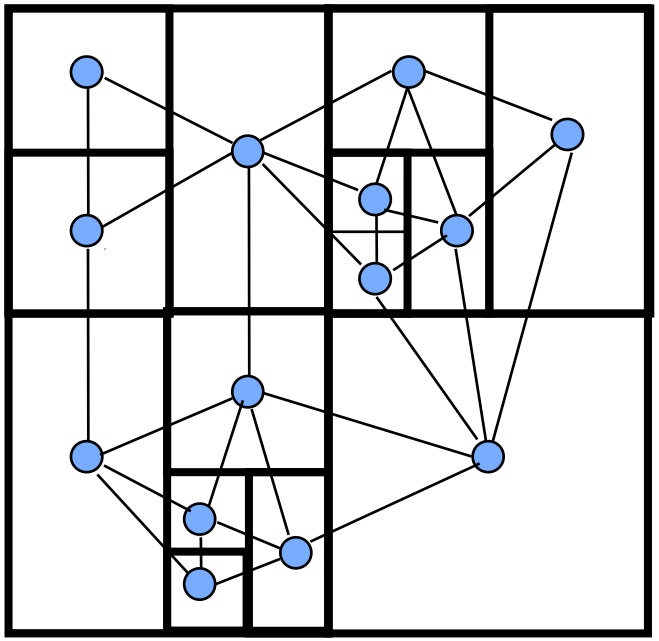
Network Structure of CAN

- **Let d be the dimension of the square, cube, hyper-cube**
  - 1: line
  - 2: square
  - 3: cube
  - 4: ...
- **Peers connect**
  - if the areas of peers share a (d-1)-dimensional area
  - e.g. for d=2 if the rectangles touch by more than a point



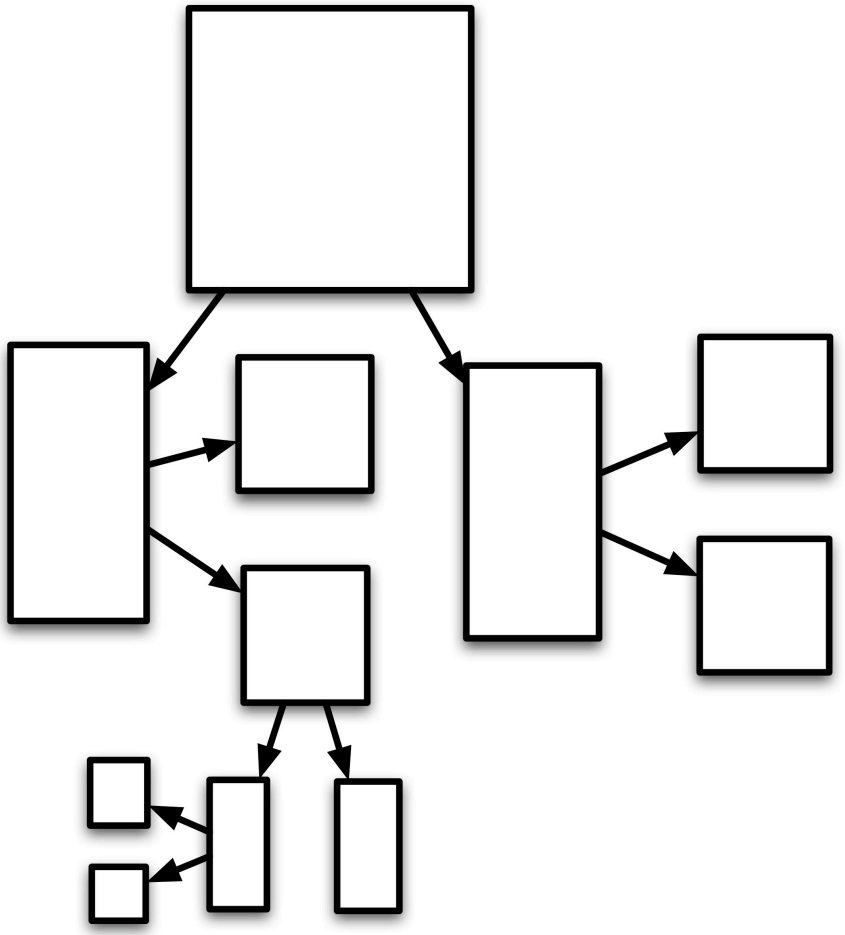
Lookup in CAN

- Compute the position of the index using the hash function on the key value
- Forward lookup to the neighbored peer which is closer to the index
- Expected number of hops for CAN in d dimensions:
  -
- Average degree of a node
  -



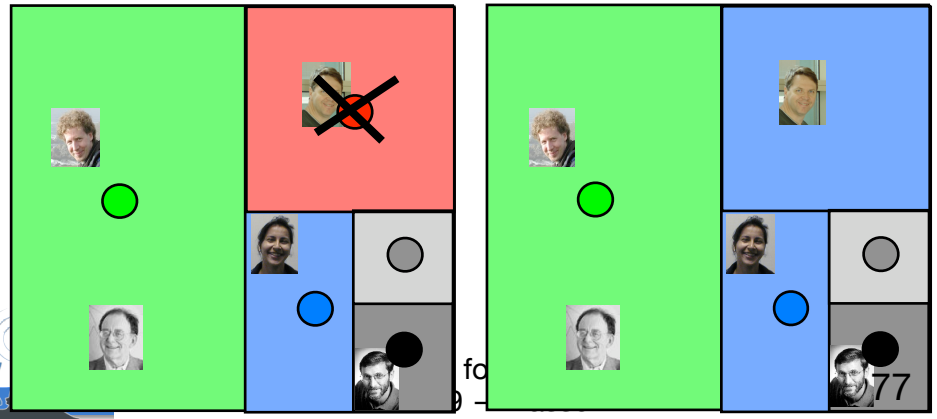
Insertions in CAN = Random Tree

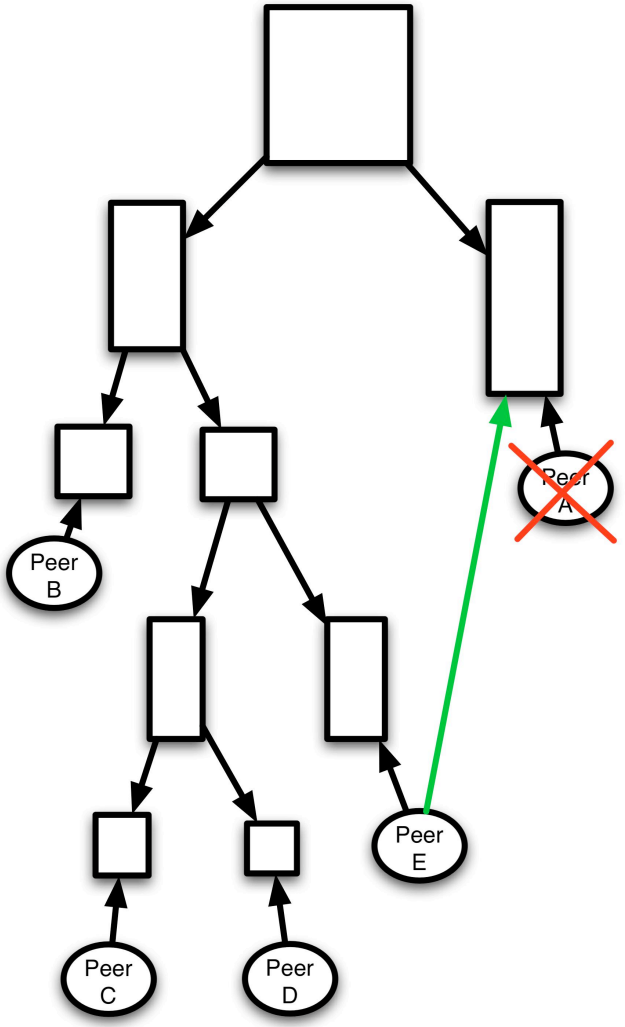
- **Random Tree**
  - new leaves are inserted randomly
  - if node is internal then flip coin to forward to left or right sub-tree
  - if node is leaf then insert two leafs to this node
- **Depth of Tree**
  - in the expectation: O(log n)
  - Depth O(log n) with high probability, i.e. $1-n^{-c}$
- **Observation**
  - CAN inserts new peers like leafs in a random tree



Leaving Peers in CAN

- **If a peer leafs**
  - he does not announce it
- **Neighbors continue testing on the neighborhood**
  - to find out whether peer has left
  - the first neighboir who finds a missing neighbor takes over the area of the missing peer
- **Peers can be responsible for many rectangles**
- **Repeated insertions and deletions of peers leed to fragmentation**

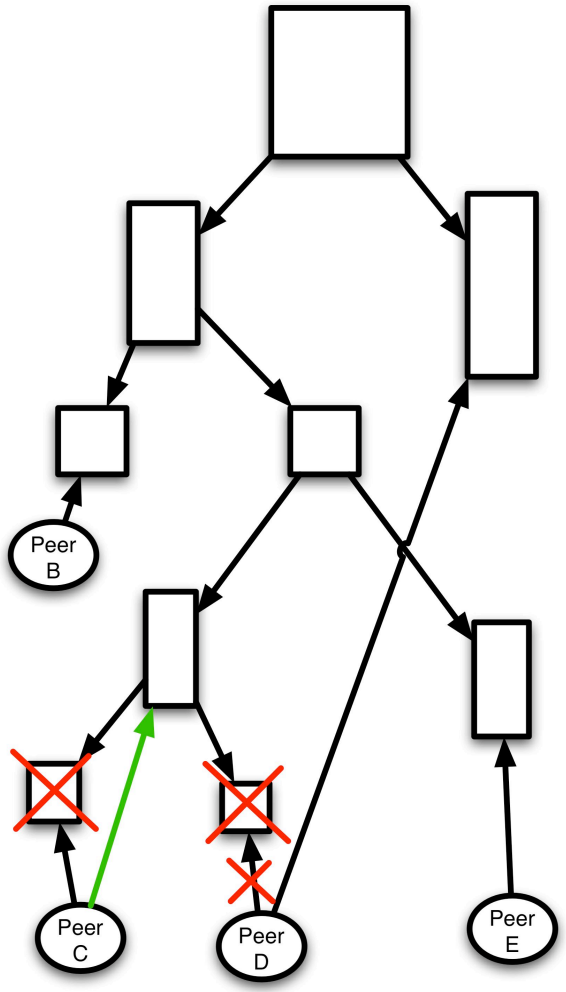




Computer Networks and Telematics
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer
CoNe
Freiburg

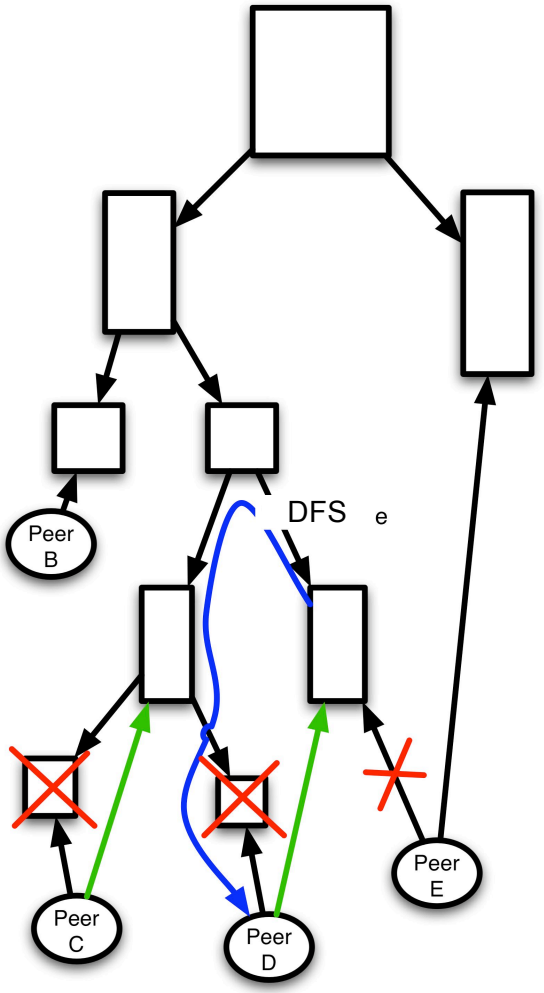
Defragmentation — The Simple Case

- **To heal fragmented areas**
  - from time to to time areas are freshly assigned
- **Every peer with at least two zones**
  - erases smalles zone
  - finds replacement peer for this zone
- **1. case: neighboring zone is undivided**
  - both peers are leafs in the random tree
  - transfer zone to the neighbor



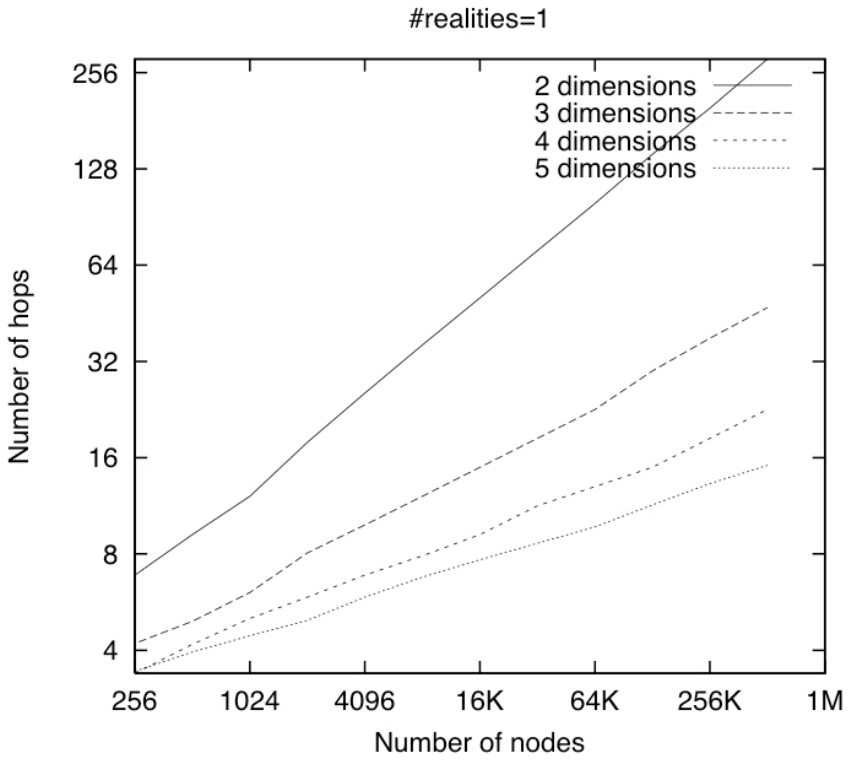
Defragmentation — The Difficult Case

- **Every peer with at least two zones**
  - erases smalles zone
  - finds replacement peer for this zone
- **2. case: neighboring zone is further divided**
  - Perform DFS (depth first search) in neighbor tree until two neighbored leafs are found
  - Transfer the zone to one leaf which gives up his zone
  - Choose the other leaf to receive the latter zone



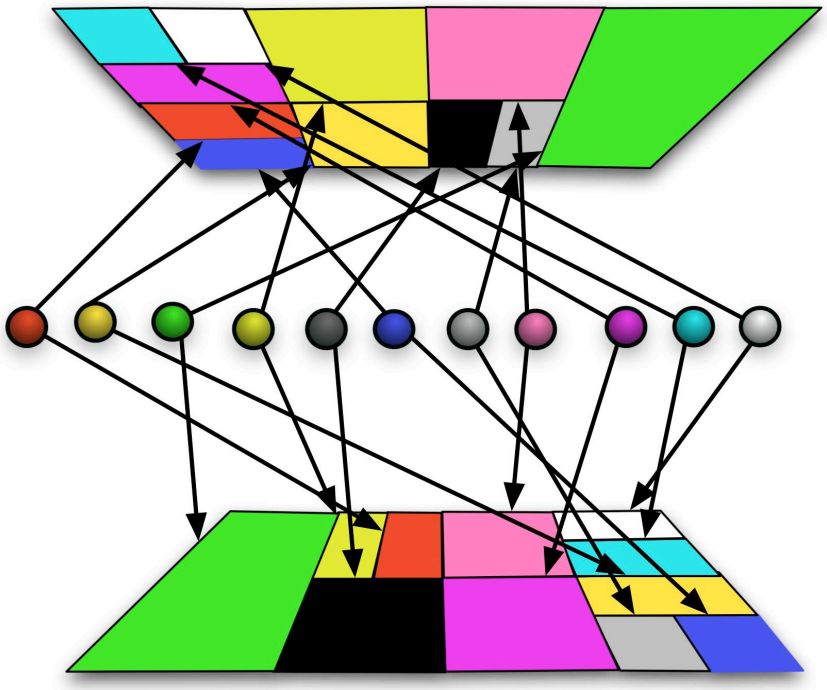
Higher Dimensions

- **Let d be the dimension of the square, cube, hyper-cube**
  - 1: line
  - 2: square
  - 3: cube
  - 4: ...
- **The expected path length is $O(n^{1/d})$**
- **Average number of neighbors O(d)**



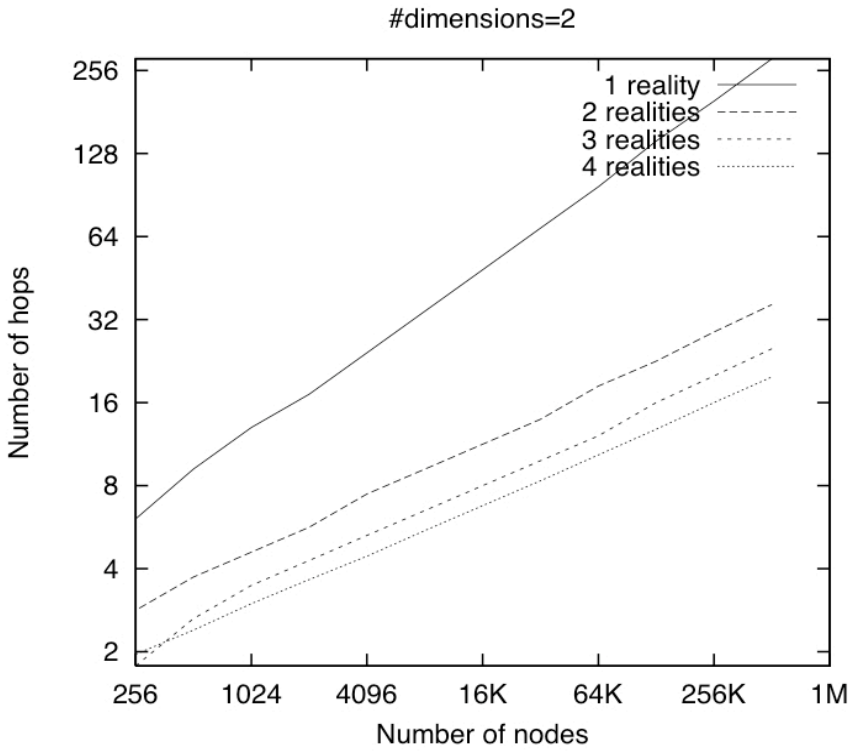
More Realities

- **Build simultanously r CANs with the same peers**
- **Each CAN is called a *reality***
- **For lookup**
  - greedily jump between realities
  - choose reality with the closest distance to the target
- **Advantanges**
  - robuster network
  - faster search



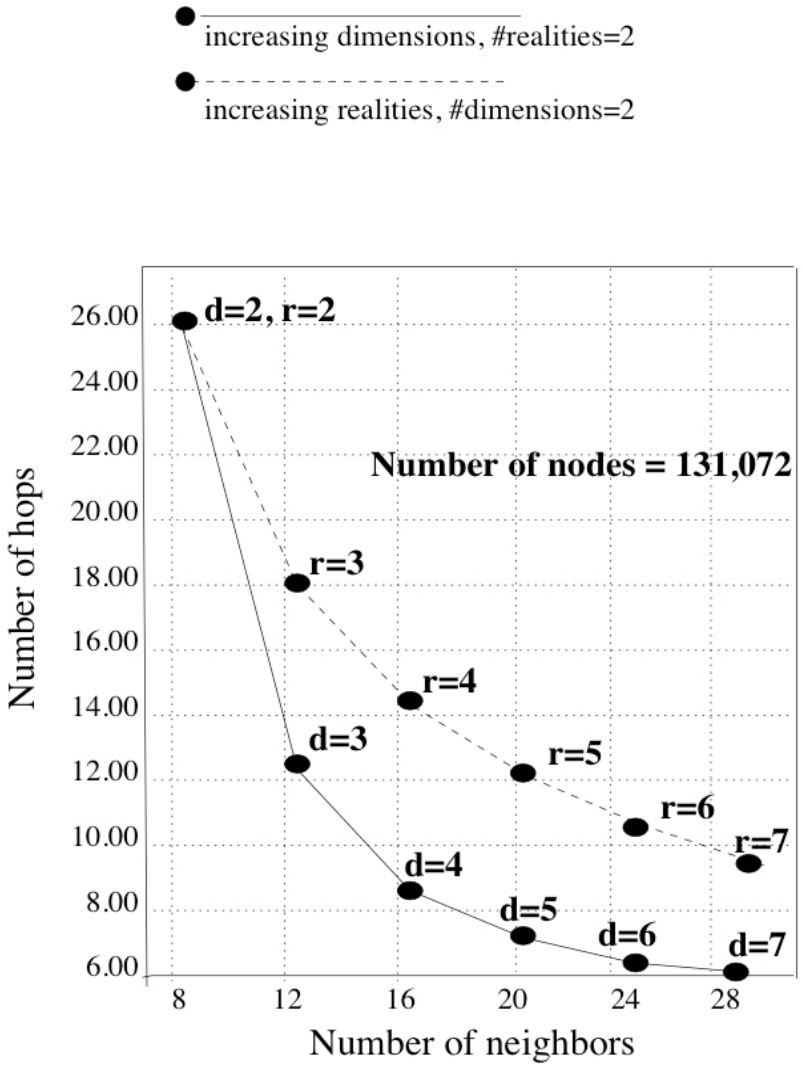
More Realities

- **Advantages**
  - robuster
  - shorter paths



Realities vs. Dimensions

- **Dimensionens reduce the lookup path length more effciently**
- **Realities produce more robust networks**

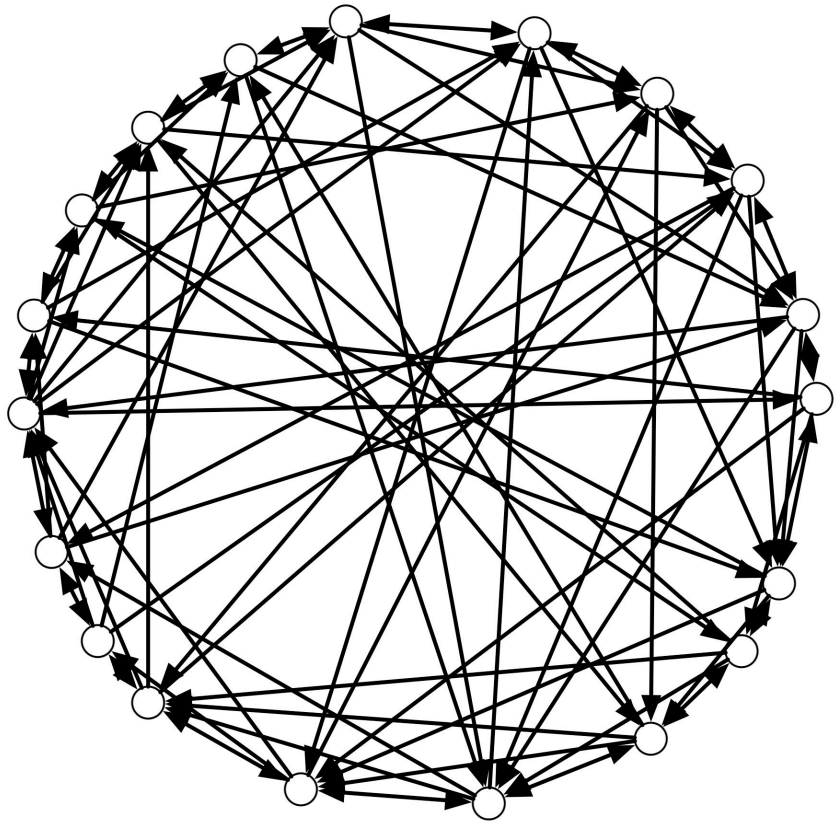


Peer-to-Peer Networks

Chord

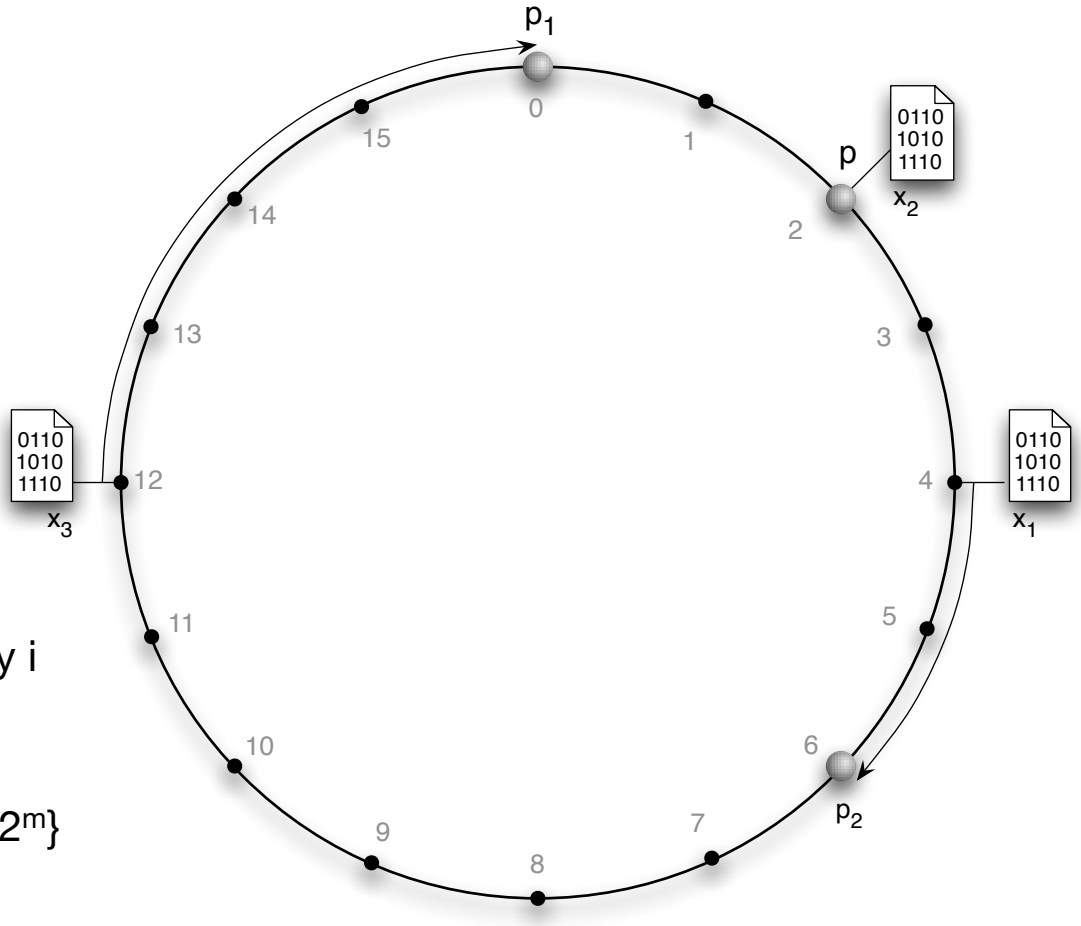
Chord

- **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



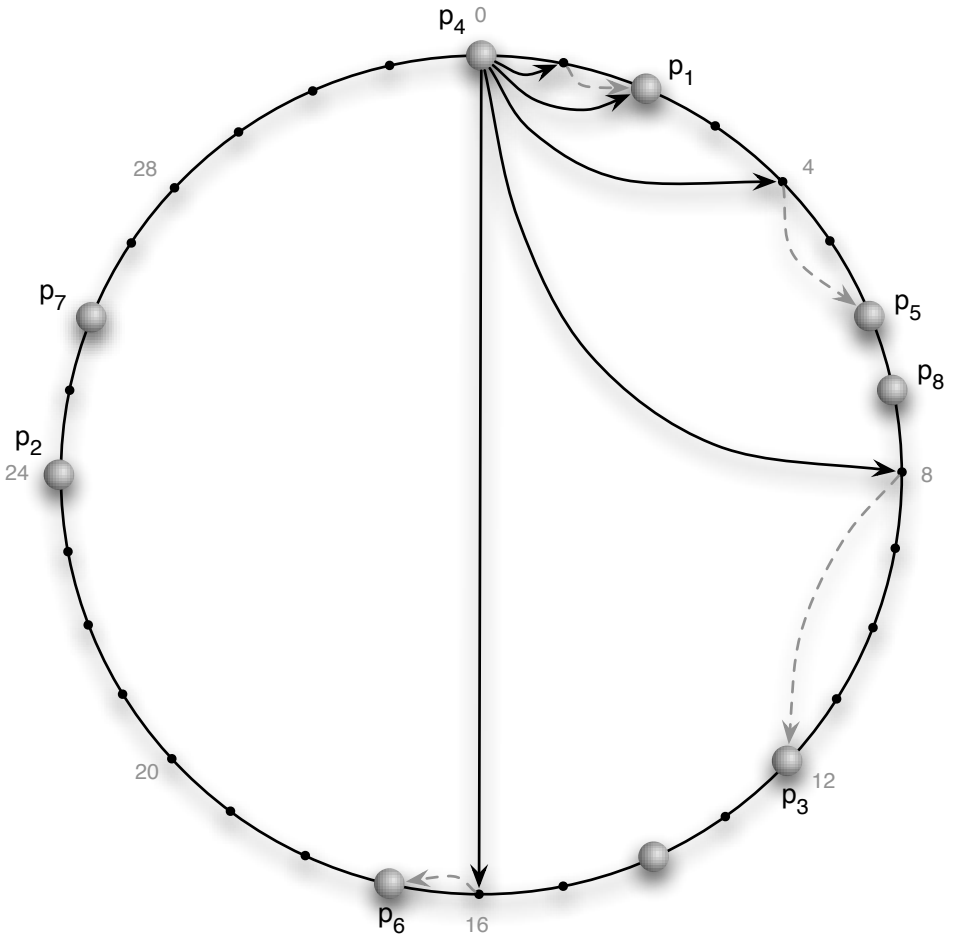
Chord as DHT

- **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 = $f_V(i)$**
  - $f_V(i) := \arg\min_{b \in V}\{(r_V(b)-r_K(i)) \bmod 2^m\}$



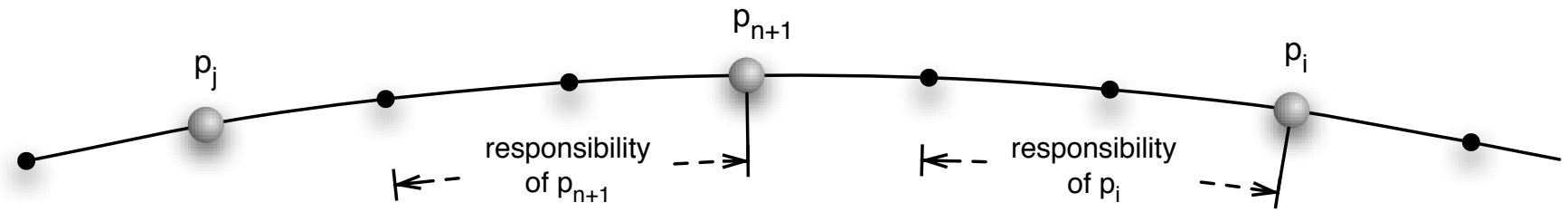
Pointer Structure of Chord

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



Balance in Chord

- **Theorem**
  - We observe in Chord for n peers and k data entries
    - Balance&Load: Every peer stores at most O(k/n log n) entries with high probability
    - Dynamics: If a peer enters the Chord then at most O(k/n log n) data entries need to be moved
- **Proof**
  - …



Properties of the DHT

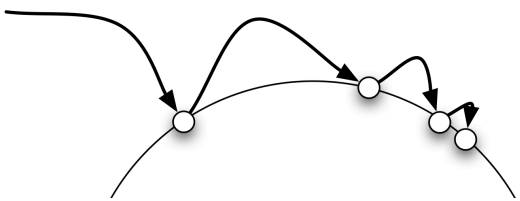
- **Lemma**
  - For all peers b the distance $|r_V(b.succ) - r_V(b)|$ is
    - in the expectation $2^m/n$,
    - $O((2^m/n) \log n)$ with high probability (w.h.p.)
    - $2^m/n^{c+1}$ für a constant $c>0$ with high probability
  - In an interval of length w $2^m/n$ we find
    - $\Theta(w)$ peers, if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

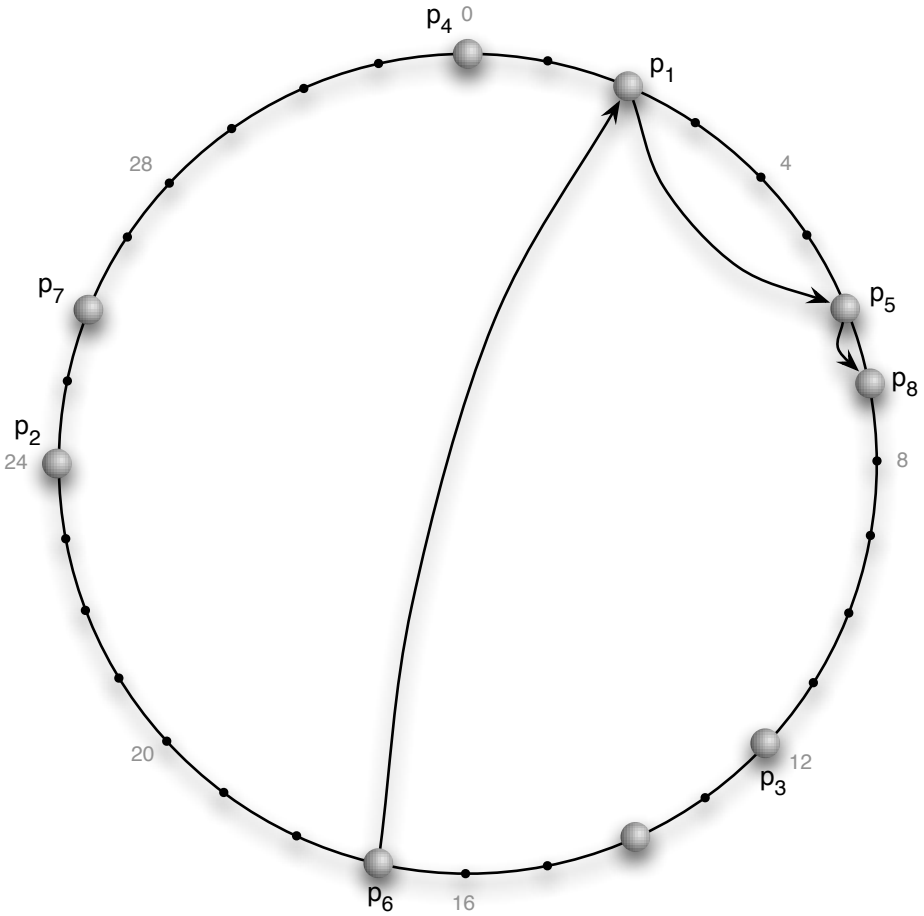
Lookup in Chord

- **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 $r_K(s) \in [r_V(b.finger[i]),r_V(finger[i+1])]$ then
          b ← b.finger[i]
        fi
      od

Lookup in Chord

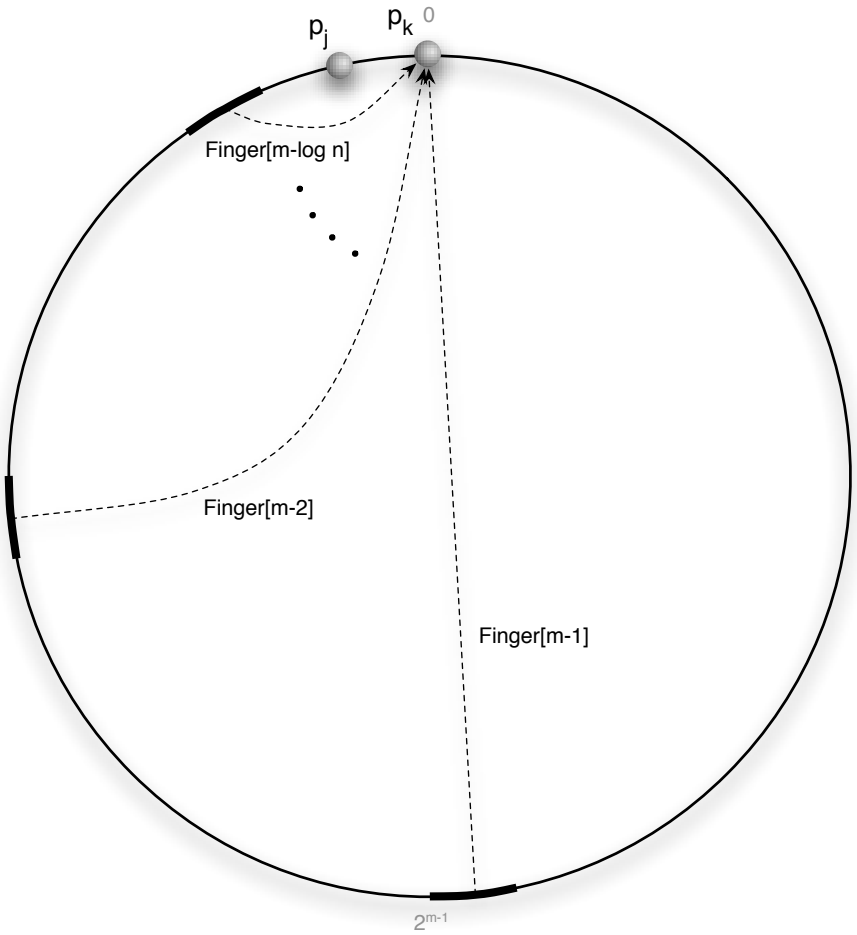
- **Theorem**
  - The Lookup in Chord needs O(log n) steps w.h.p.
- **Proof:**
  - Every hops at least halves the distance to the target
  - At the beginning the distance is at most
  - The minimum distance between is $2^m/n^c$ w.h.p.
  - Hence, the runtime is bounded by c log n w.h.p.





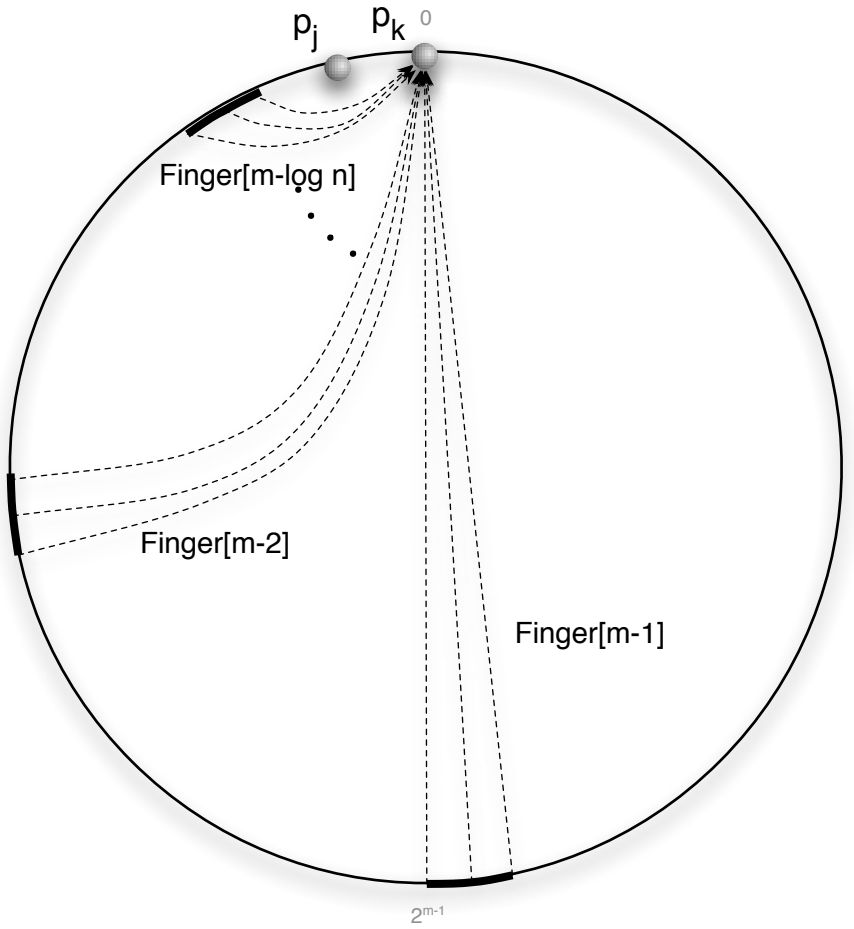
How Many Fingers?

- **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.
- **Proof**
  - The minimum distance between peers is $2^m/n^c$ w.h.p.
    - this implies that that the out-degree is O(log n) w.h.p.
  - The maximum distance between peers is O(log n $2^m/n$) w.h.p.
    - the overall length of all line segments where peers can point to a peer following a maximum distance is $O(\log^2 n\ 2^m/n)$
    - in an area of size $w=O(\log^2 n)$ there are at most $O(\log^2 n)$ w.h.p.



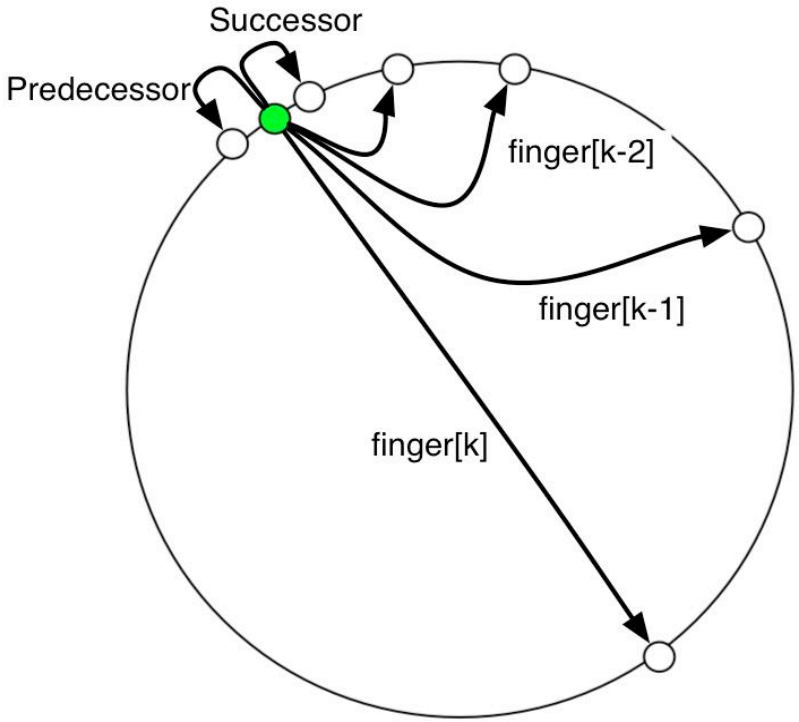
Adding a Peer

- **Theorem**
  - For integrating a new peer into Chord only O(log2 n) messages are necessary.
- **Algorithm**
  - Find the target area in O(log n) steps
  - Outgoing O(log n) pointers adopted from the predecessor and successor
  - The in-degree of the new peer is $O(\log^2 n)$ w.h.p.



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

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

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**

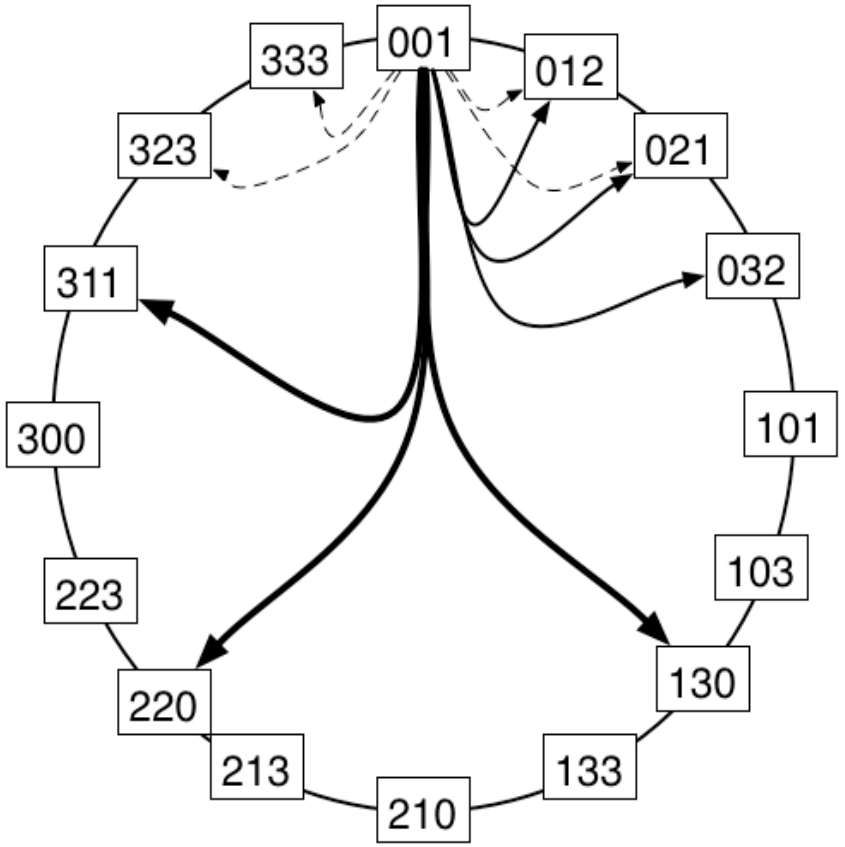
Routing Table

- **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**
  - ℓ/2 with next higher ID
  - ℓ/2 with next lower ID

0	1	2	3	4	5		7	8	9	a	b	c	d	e	f
x	x	x	x	x	x		x	x	x	x	x	x	x	x	x
6	6	6	6	6		6	6	6	6	6	6	6	6	6	6
0	1	2	3	4		6	7	8	9	a	b	c	d	e	f
x	x	x	x	x		x	x	x	x	x	x	x	x	x	x
6	6	6	6	6	6	6	6	6	6		6	6	6	6	6
5	5	5	5	5	5	5	5	5	5		5	5	5	5	5
0	1	2	3	4	5	6	7	8	9		b	c	d	e	f
x	x	x	x	x	x	x	x	x	x		x	x	x	x	x
6		6	6	6	6	6	6	6	6	6	6	6	6	6	6
5		5	5	5	5	5	5	5	5	5	5	5	5	5	5
a		a	a	a	a	a	a	a	a	a	a	a	a	a	a
0		2	3	4	5	6	7	8	9	a	b	c	d	e	f
x		x	x	x	x	x	x	x	x	x	x	x	x	x	x

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

$$2^{-bm}$$

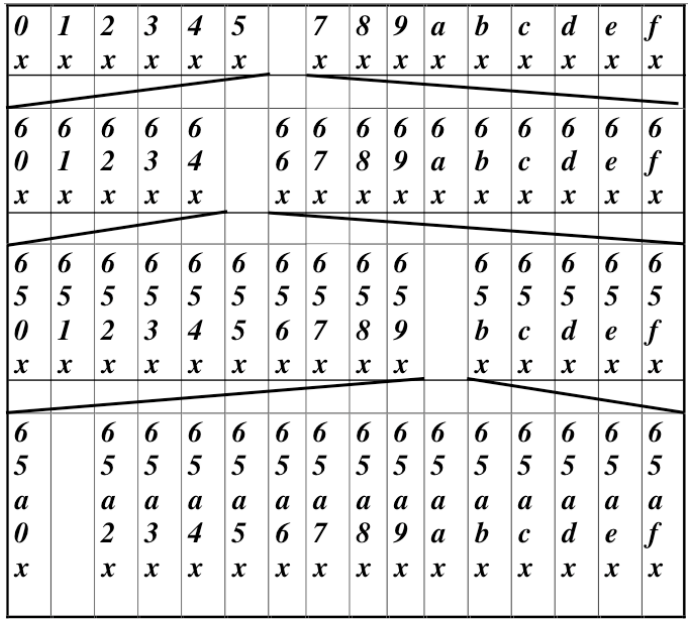
  - The probability that a m-digit prefix is unused is

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

  - For m=c (log n)/b we get

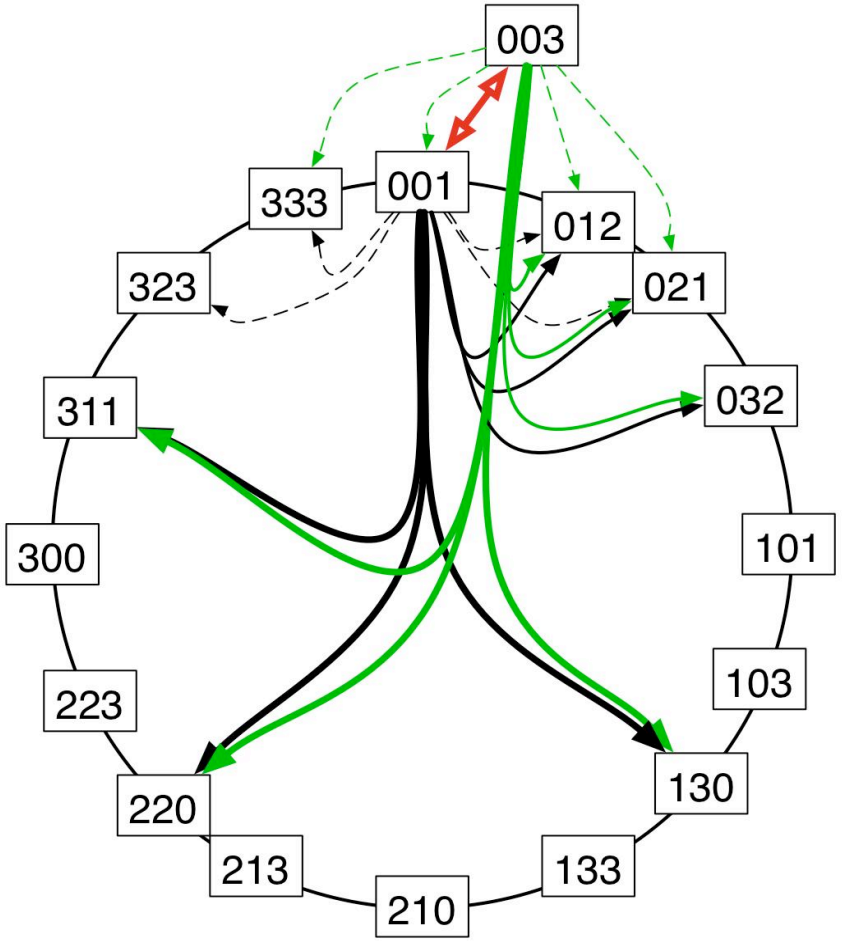
$$e^{-n/2^{bm}} \leq e^{-n/2^{c \log n}} \leq e^{-n/n^c} \leq e^{-n^{c-1}}$$

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



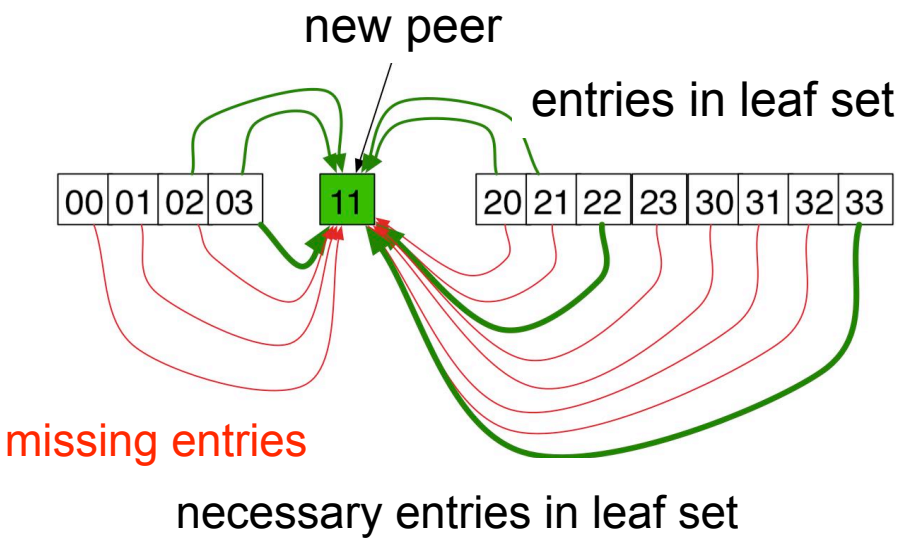
A Peer Enters

- **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 informiert 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



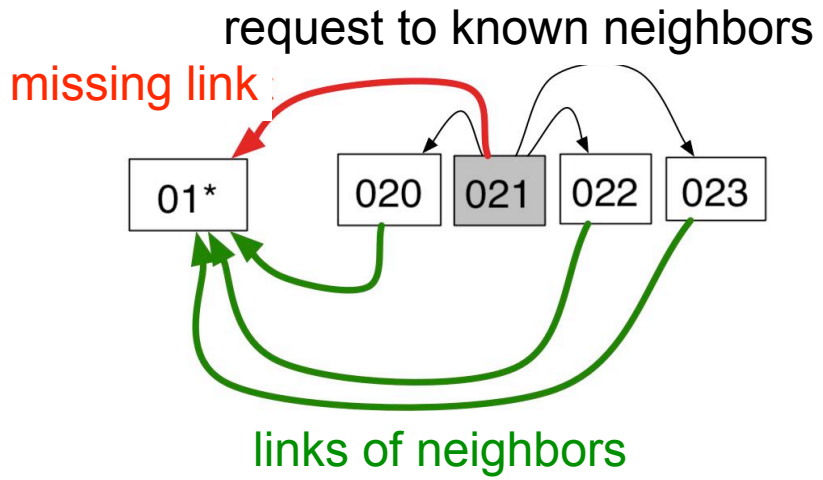
When the Entry-Operation Errs

- **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**



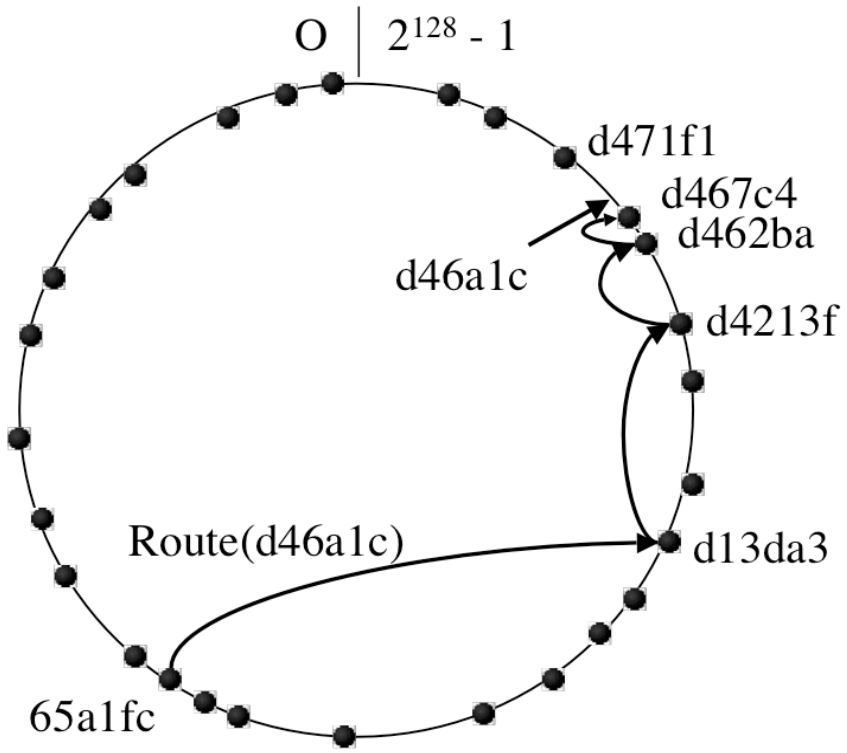
Missing Entries in the Routing Table

- **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 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**



Lookup in Detail

- **L:** ℓ-leafset
- **R:** routing table
- **M:** nodes in the vicinity of D (according to RTT)
- **D:** key
- **A:** nodeID of current peer
- **R^i_j:** 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 larges common prefix of A and D (shared header length)

```
(1)  if (L_{-⌊|L|/2⌋} ≤ D ≤ L_{⌊|L|/2⌋}) {
(2)      // D is within range of our leaf set
(3)      forward to L_i, s.th. |D − L_i| is minimal;
(4)  } else {
(5)      // use the routing table
(6)      Let l = shl(D, A);
(7)      if (R_l^{D_l} ≠ null) {
(8)          forward to R_l^{D_l};
(9)      }
(10)     else {
(11)         // rare case
(12)         forward to T ∈ L ∪ R ∪ M, s.th.
(13)             shl(T, D) ≥ l,
(14)             |T − D| < |A − D|
(15)     }
(16) }
```

Routing — Discussion

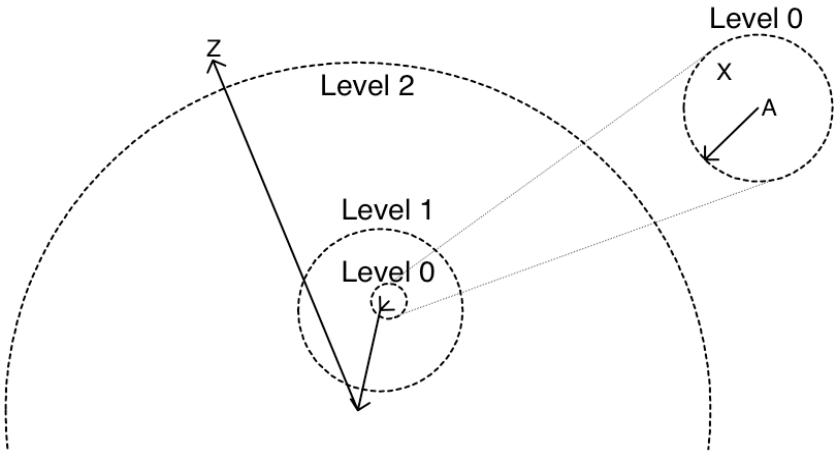
- **If the Routing-Table is correct**
  - routing needs O((log n)/b) messages
- **As long as the leaf-set is correct**
  - routing needs O(n/l) messages
  - unrealistic worst case since even damaged routing tables allow dramatic speedup
- **Routing does not use the real distances**
  - M is used only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

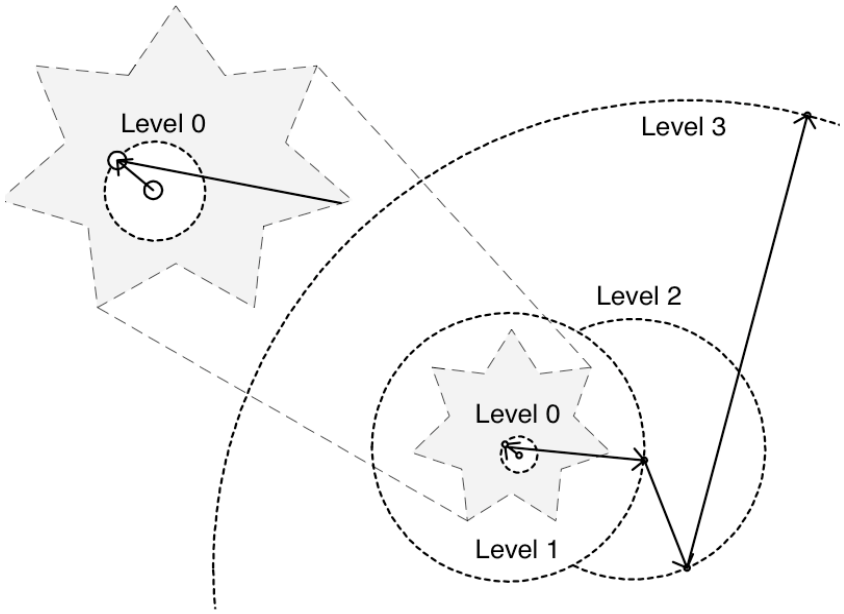
Locality in the Routing Table

- **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

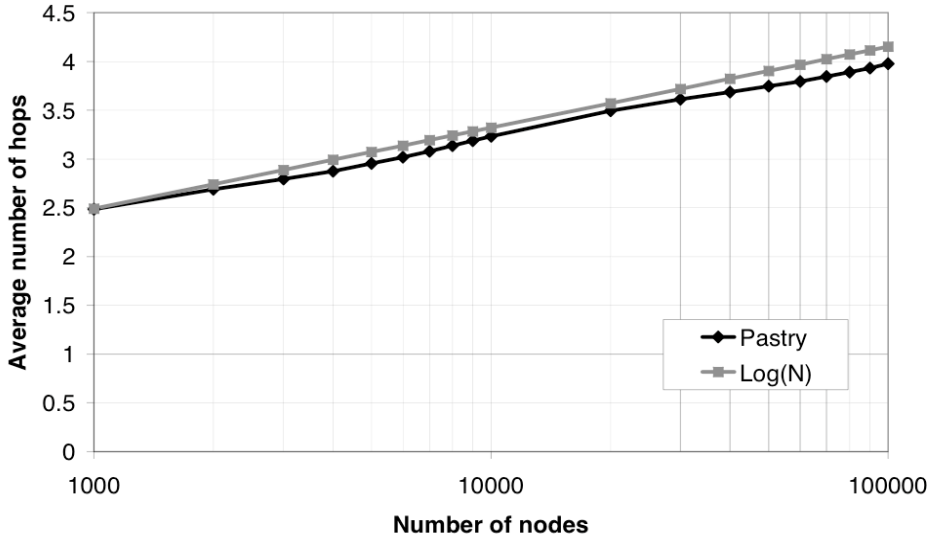


Localization of Near Nodes

- 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

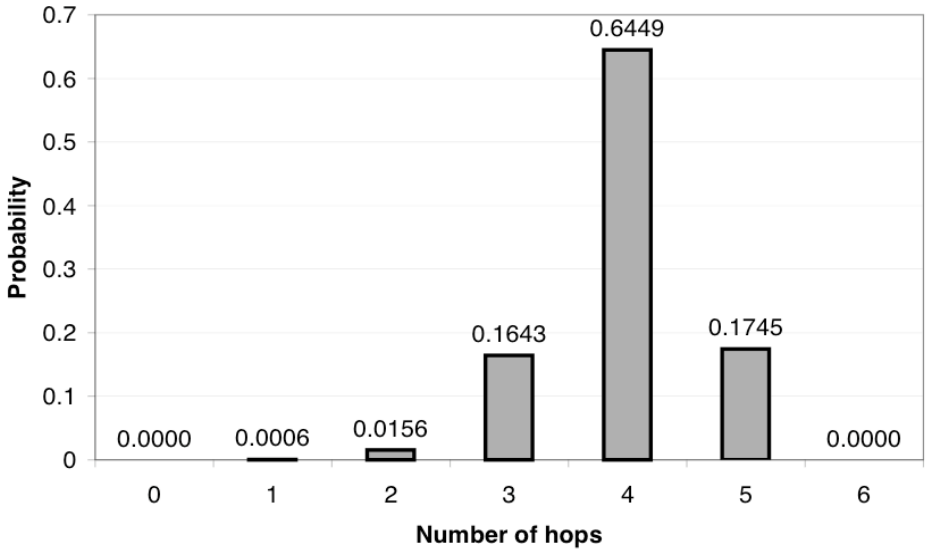
Experimental Results — Scalability

- 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



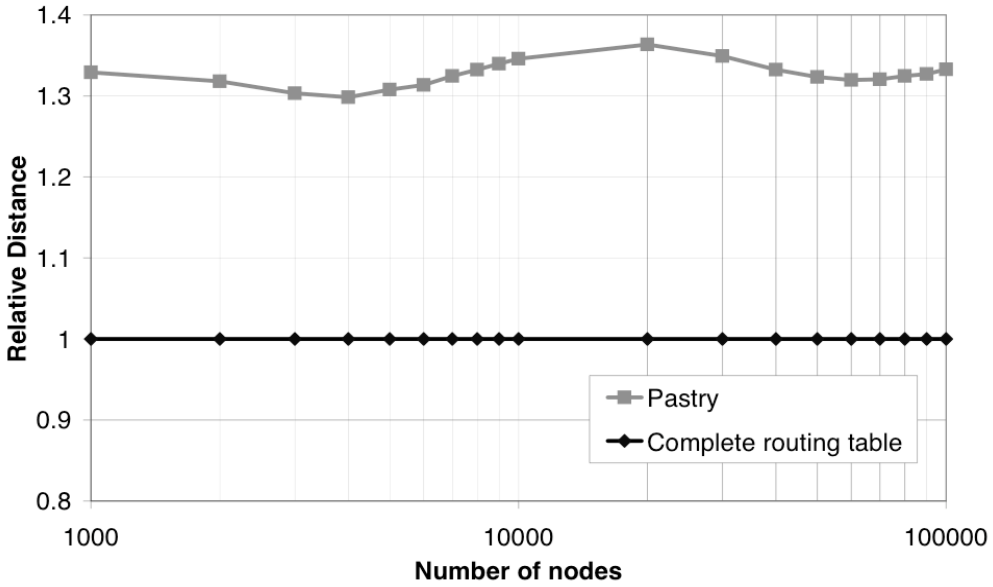
Experimental Results
Distribution of Hops

- **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



Peer-to-Peer Networks

Tapestry

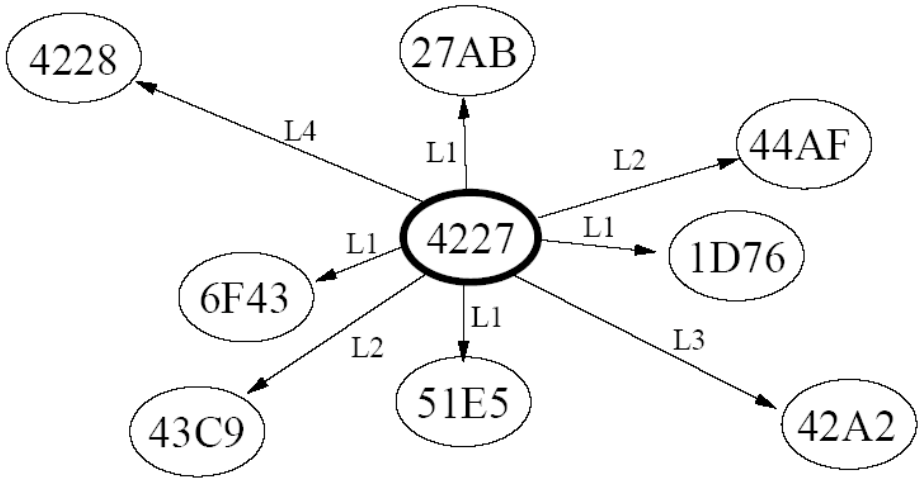
Zhao, Kubiatowicz und Joseph (2001)

Tapestry

- **Objects and Peers are identified by**
  - Objekt-IDs (Globally Unique Identifiers GUIDs) and
  - Peer-IDs
- **IDs**
  - are computed by hash functions
    - like CAN or Chord
  - are strings on basis B
    - B=16 (hexadecimal system)

Neighborhood of a Peer (1)

- **Every peer A maintains for each prefix x of the Peer-ID**
  - if a link to another peer sharing this Prefix x
  - i.e. peer with ID B=xy has a neighbor A, if xy´=A for some y, y´
- **Links sorted according levels**
  - the level denotes the length of the common prefix
  - Level L = |x|+1

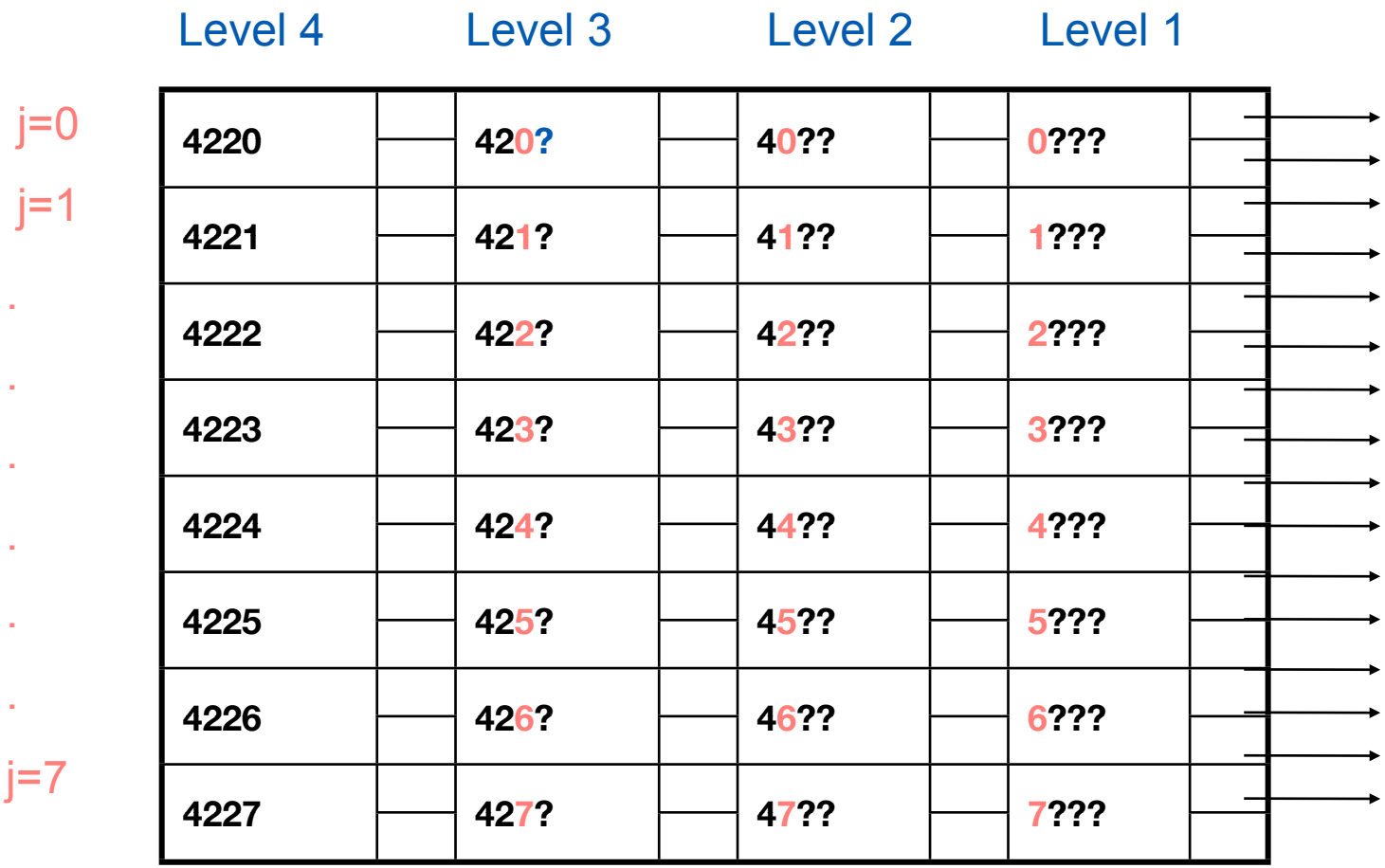


Neighborhood Set (2)

- **For each prefix x and all letters j of the peer with ID A**
  - establish a link to a node with prefix xj within the neighboorhood set $N^A_{x,j}$
- **Peer with Node-ID A has b |A| neighborhood sets**
- **The neighborhood set of contains all nodes with prefix sj**
  - Nodes of this set are denoted by (x,j)

Example of Neighborhood Sets

Neighborhood set of node 4221



Links

- **For each neighborhood set at most k Links are maintained**

$$k \geq 1 : \left| N_{x,j}^{A} \right| \leq k$$

- **Note:**
  - some neighborhood sets are empty

Properties of Neighborhood Sets

- **Consistency**
  - If $N_{x,j}^{A} = \emptyset$ für any A
    - then there are no (x,j) peers in the network
    - this is called a hole in the routing table of level |x|+1 with letter j
- **Network is always connected**
  - Routing can be done by following the letters of the ID $b_1 b_2 \ldots b_n$

N_{ϕ,b_1}^{A} 1st hop to node A_1

$N_{b_1,b_2}^{A_1}$ 2nd hop to node A_2

$N_{b_1 o b_2,b_3}^{A_2}$ 3rd hop to node A_3

…

Locality

- **Metric**
  - e.g. given by the latency between nodes
- **Primary node of a neighborhood set** $N^A_{x,j}$
  - The closest node (according to the metric) in the neighborhood set of A is called the primary node
- **Secondary node**
  - the second closest node in the neighborhood set
- **Routing table**
  - has primary and secondary node of the neighborhood table

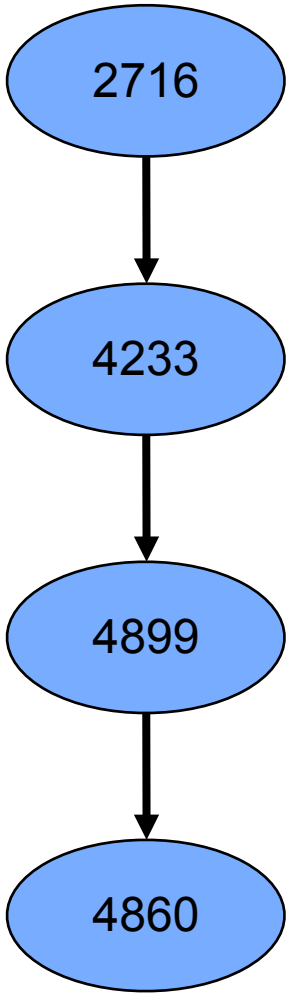
Root Node

- Object with ID Y should stored by a so-called Root Node with this ID
- If this ID does not exist then a deterministic choice computes the next best choice sharing the greatest common prefix using Surrogate Routing

Surrogate Routing

- **Surrogate Routing**
  - compute a surrogate (replacement root node)
  - If (x,j) is a hole, then choose (x,j+1),(x,j+2),… until a node is found
  - Continue search in the next higher level

Example: Surrogate Routing



- **Lookup of 4666 by peer 2716**

Level 1, j=4

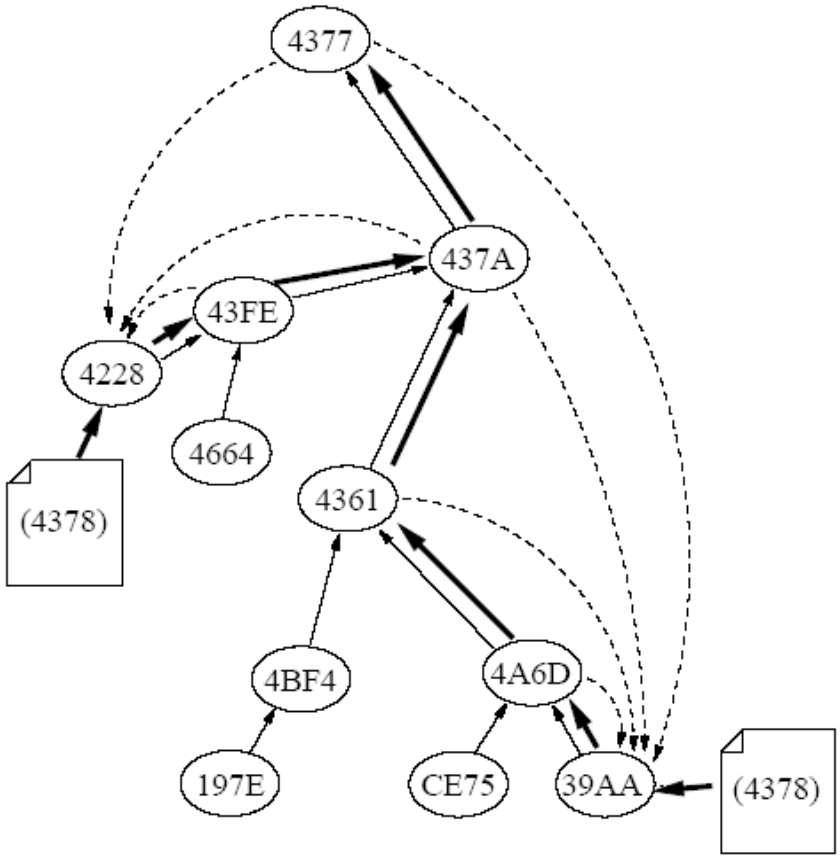
Level 2, j=6 does not exist, next link j=8

Level 3, j=6

Peer 4860 has no level 4 neighbors => end of search

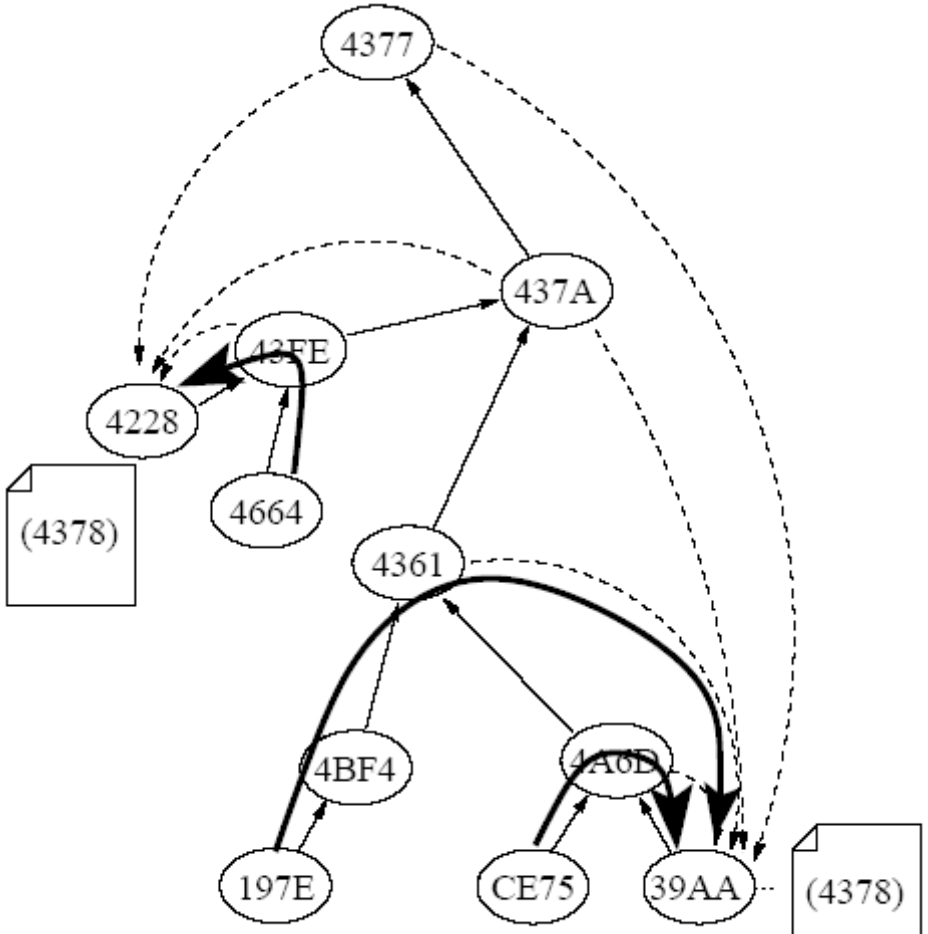
Publishing Objects

- **Peers offering an object (storage servers)**
  - send message to the root node
- **All nodes along the search path store object pointers to the storage server**



Lookup

- **Choose the root node of Y**
- **Send a message to this node**
  - using primary nodes
- **Abort search if an object link has been found**
  - then send message to the storage server



Fault Tolerance

- **Copies of object IDs**
  - use different hash functions for multiple root nodes for objects
  - failed searches can be repeated with different root nodes
- **Soft State Pointer**
  - links of objects are erased after a designated time
  - storage servers have to republish
    - prevents dead links
    - new peers receive fresh information

Surrogate Routing

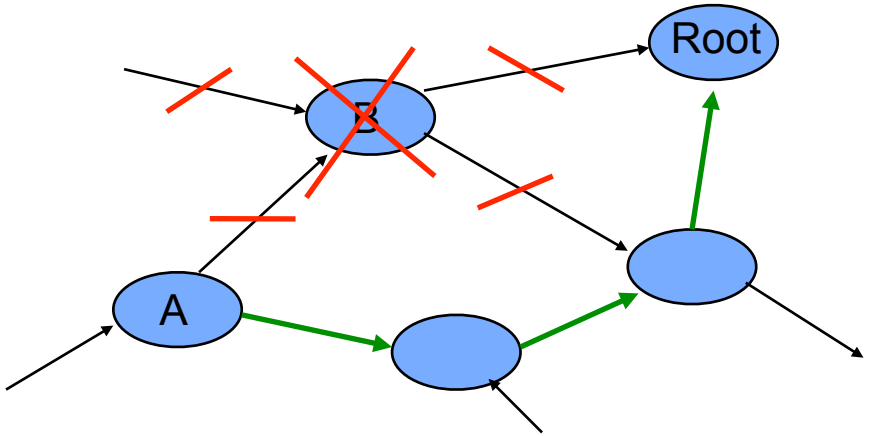
- **Theorem**
  - Routing in Tapestry needs O(log n) hops with high probability

Adding Peers

- **Perform lookup in the network for the own ID**
  - every message is acknowledged
  - send message to all neighbors with fitting prefix,
    - Acknowledged Multicast Algorithm
- **Copy neighborhood tables of surrogate peer**
- **Contact peers with holes in the routing tables**
  - so they can add the entry
  - for this perform multicast algorithm for finding such peers

Leaving of Peers

- **Peer A notices that peer B has left**
- **Erase B from routing table**
  - Problem holes in the network can occur
- **Solution: Acknowledged Multicast Algorithm**
- **Republish all object with next hop to root peer B**



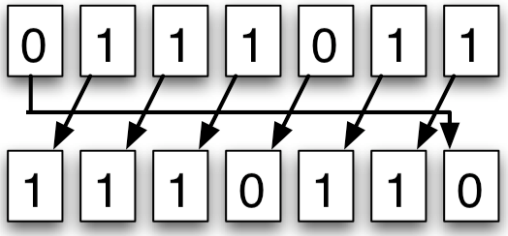
Peer-to-Peer Networks

Koorde

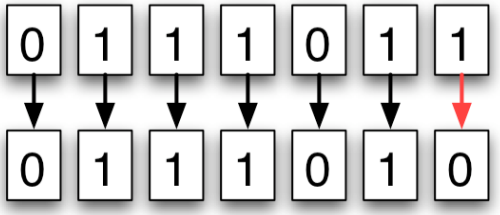
Shuffle, Exchange, Shuffle-Exchange

- **Shuffle-Operation:**
  - shuffle(s_1, s_2, s_3,..., s_m) = (s_2,s_3,..., s_m,s_1)
  - Exchange:
    - exchange(s_1, s_2, s_3,..., s_m) = (s_1, s_2, s_3,..., $\neg s_m$)
  - Shuffle-Exchange:
    - SE(S) = exchange(shuffle(S)) = (s_2,s_3,..., s_m, $\neg$ s_1)

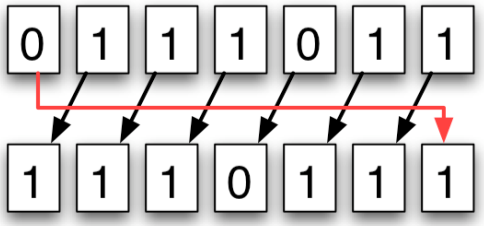
Shuffle



Exchange



Shuffle-Exchange



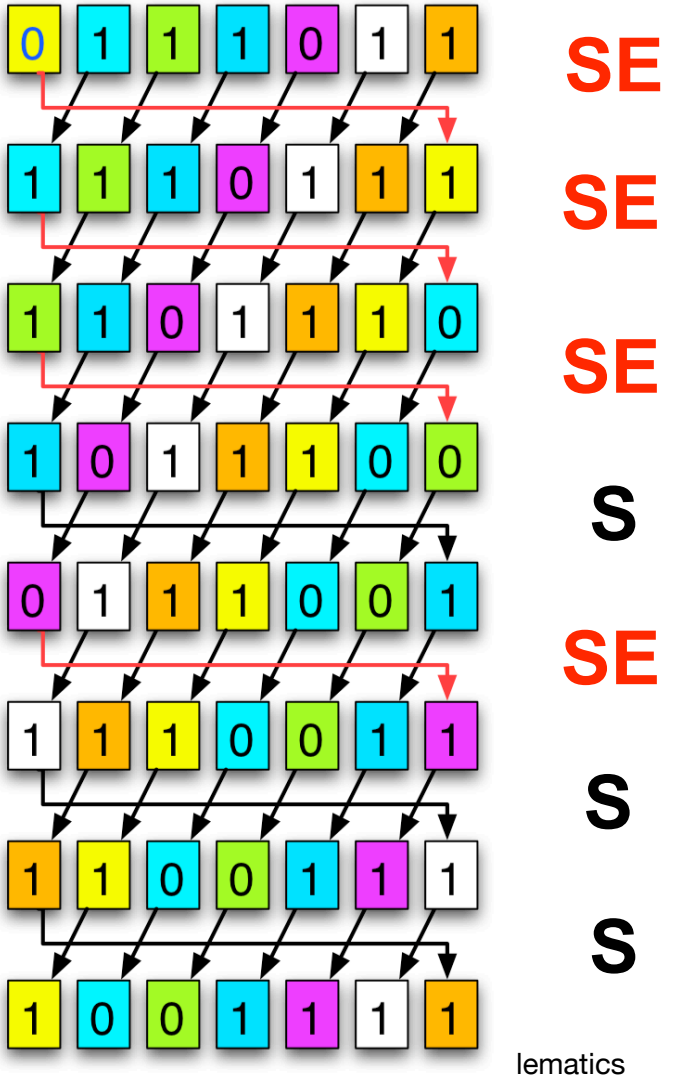
Do the Shuffle

- Routing

Example:

From	0	1	1	1	0	1	1
To	1	0	0	1	1	1	1
use	SE	SE	SE	S	SE	S	S

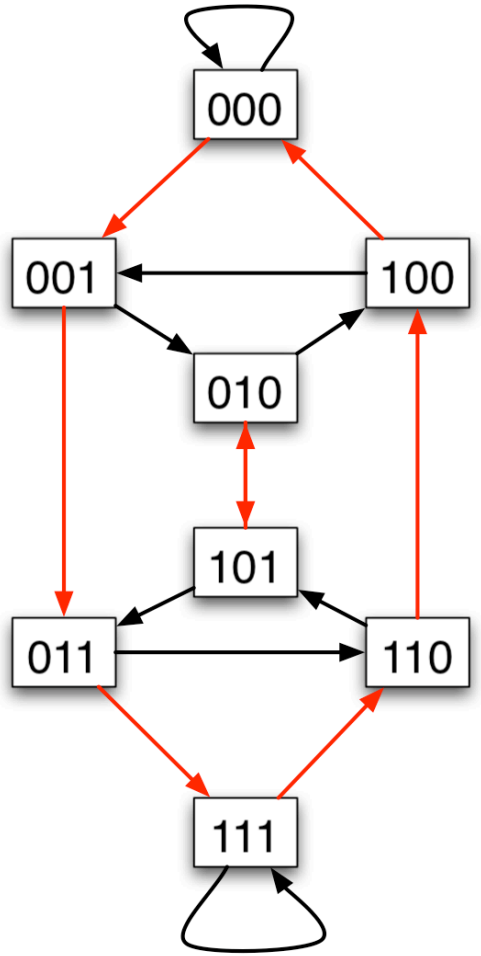
operations



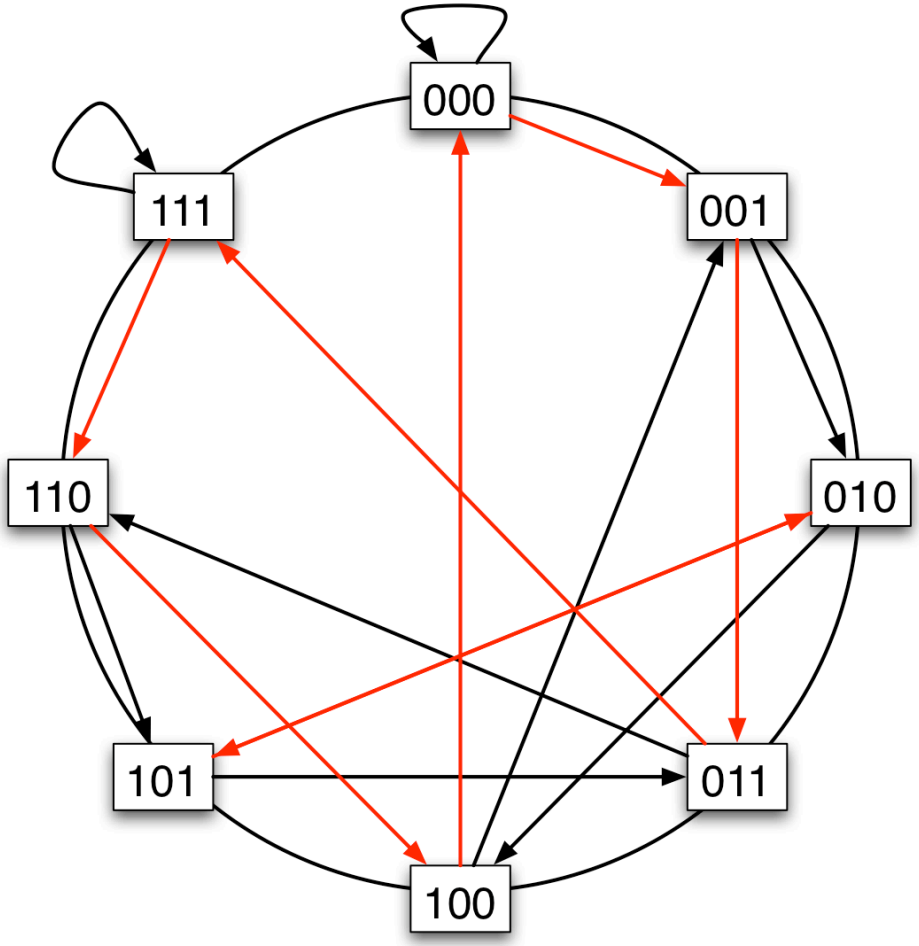
DeBruijn-Graph

- **$n=2^m$ nodes**
- **Edges:**
  - (u,shuffle(u))
  - (u,SE(u))

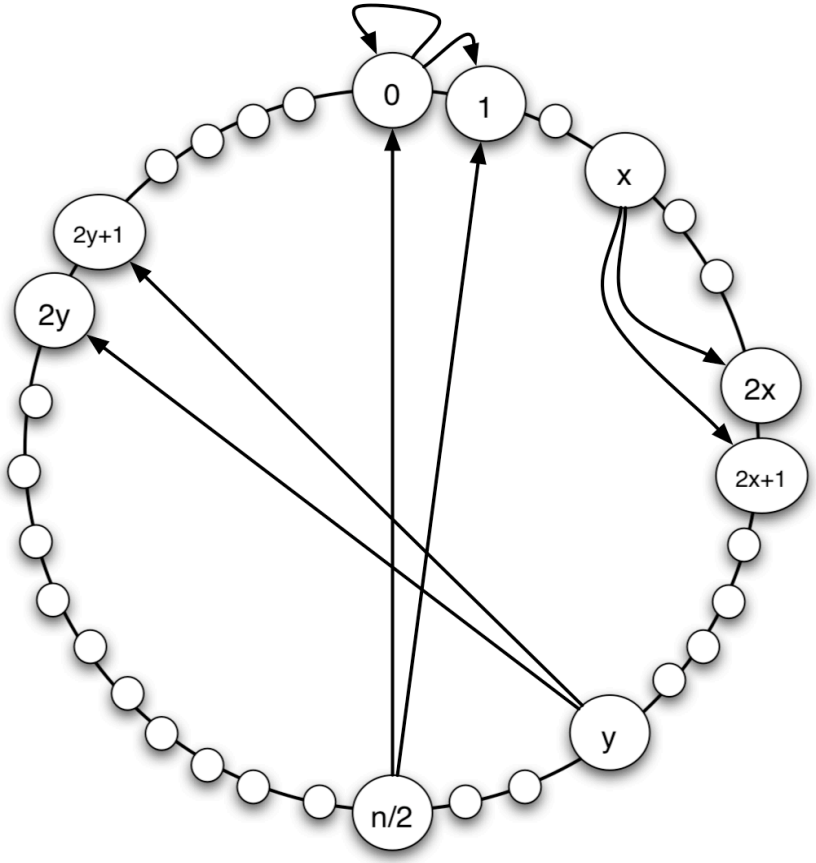
- **Lemma**
  - DeBruijn-Graph has degree 2 and diameter log n
- **Koorde = Ring + DeBruijn-Graph**



Koorde = Ring + DeBruijn-Graph

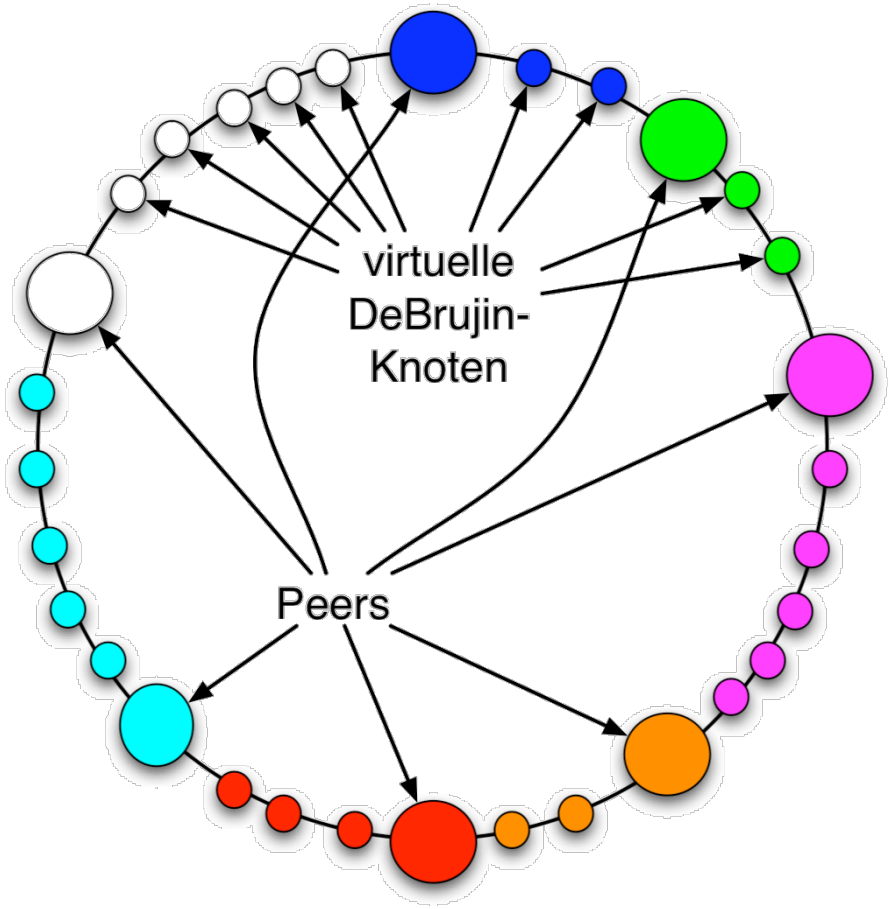


Koorde = Ring + DeBruijn-Graph



Virtual DeBruijn Nodes

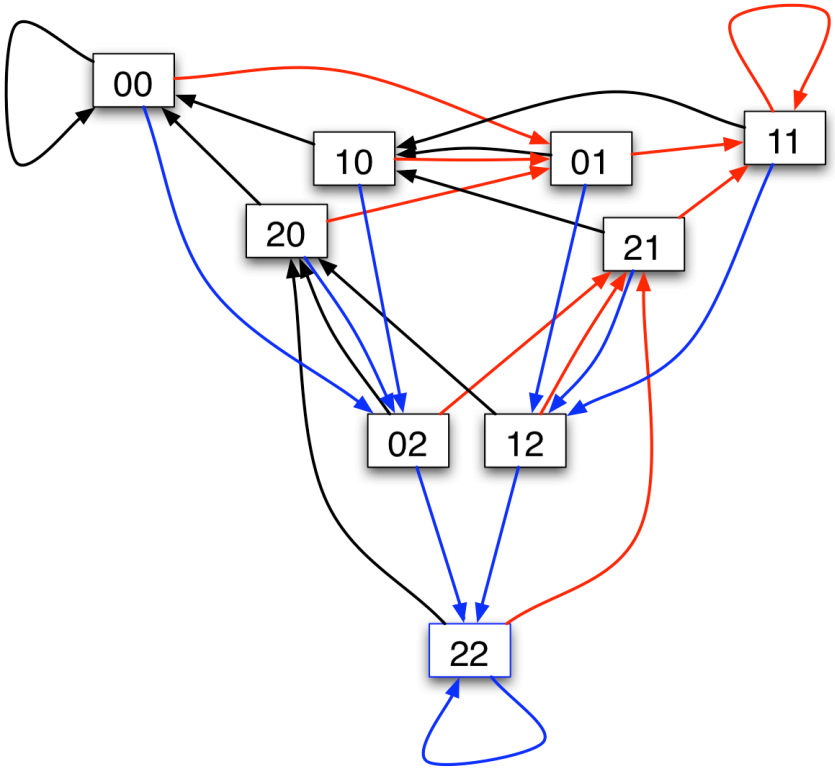
- **Choose m > (1+c) log (n) to avoid collisions**
- **Problem:**
  - more nodes than pees
- **Solution:**
  - Every peer manages an interval of DeBruijn-nodes for incoming edges
  - for outgoing edges consider only the original edges.



Properties of Koorde

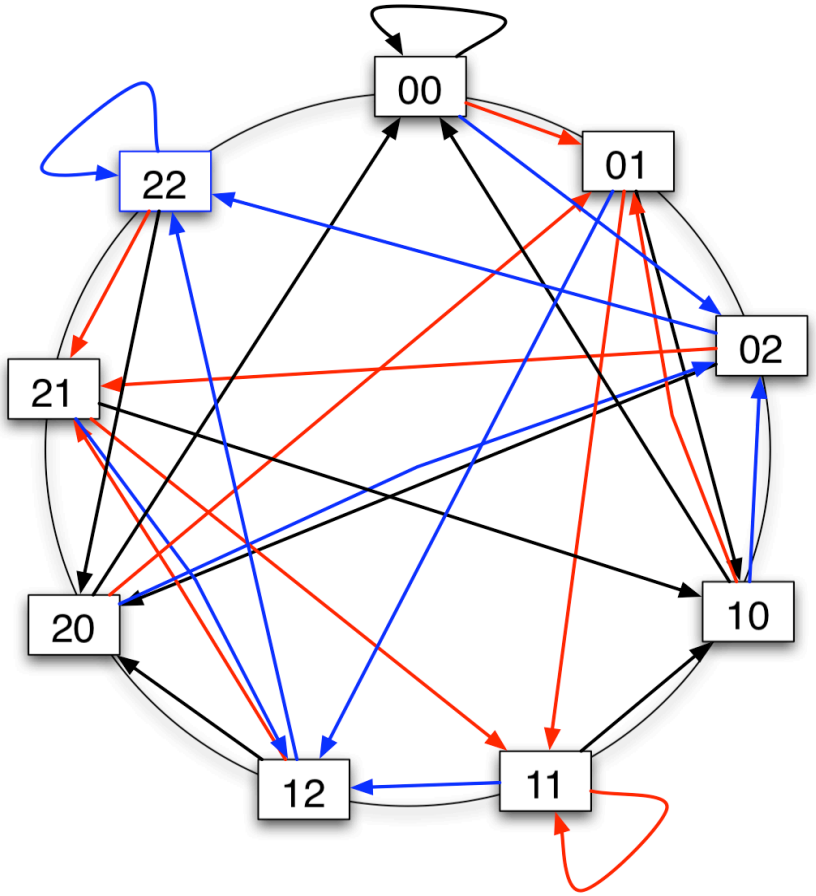
- **Theorem**
  - Every node has at most four pointers
  - The indegree is at most O(log n) w.h.p.
  - The diameter an the lookup time is O(log n) w.h.p.

Extension: Degree-k-DeBruijn-Graph



k-Koorde

- **Search time and diameter O((log n)/(log k))**

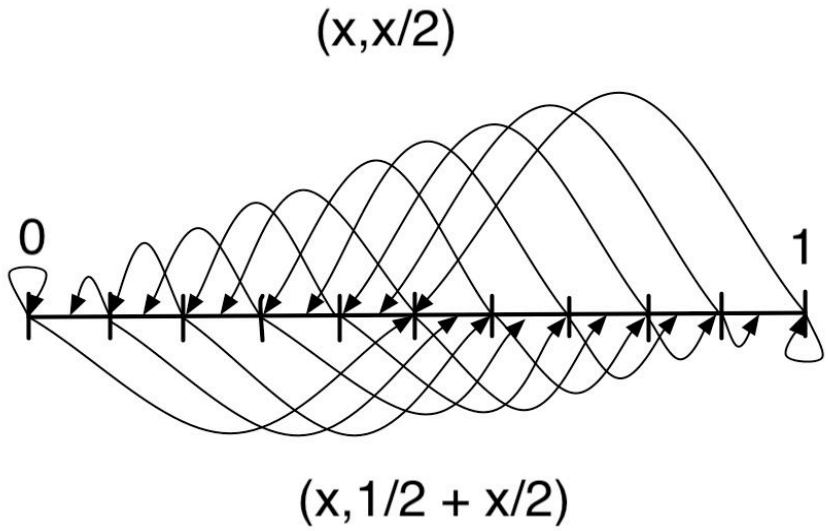


Peer-to-Peer Networks

Distance Halving

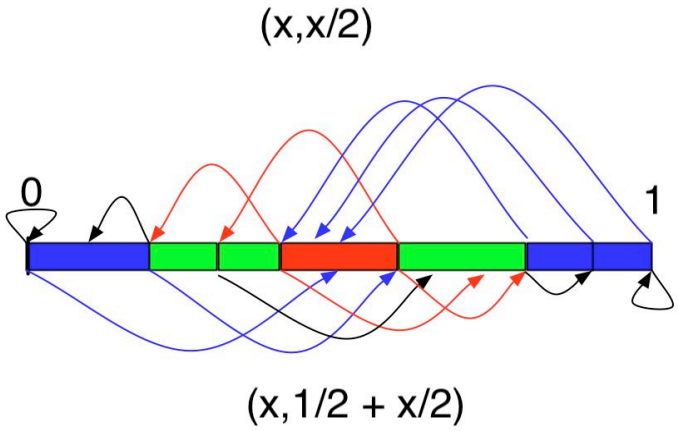
Continuous Graphs

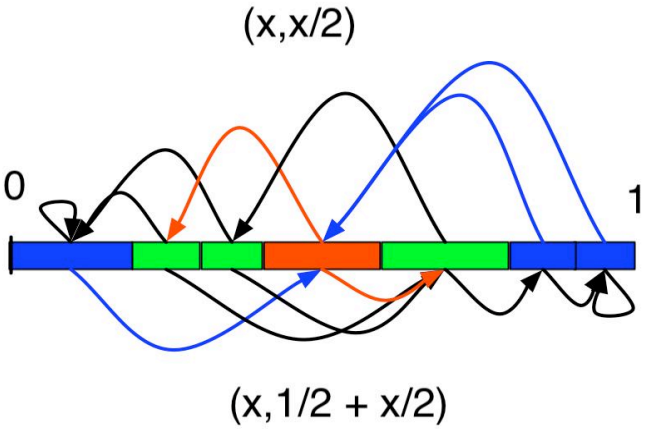
- **Graphs with infinite number of nodes and edges**
- **Graph**
  - $x \in [0,1)$
  - Edges:
    - (x,x/2) - left edge
    - (x,1+x/2) - right edge
  - And opposite edges
    - (x/2,x)
    - (1+x/2,x)



Transition from Continuous to the Discrete Case

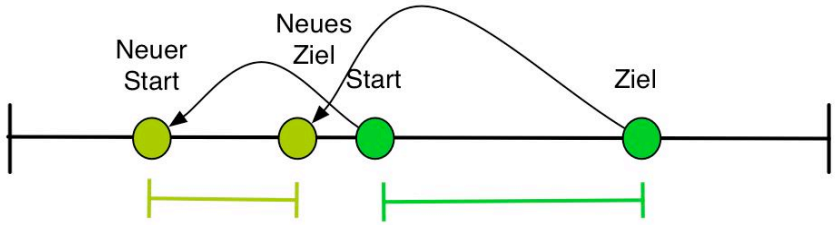
- **Consider intervals as node set**
- **Insert edge between A and B,**
  - if there is x ∈ A and y ∈ B such that (x,y) is an edge in the continuous graph
- **Intervals are added by halving existing intervals**
- **Using the multiple choice principle the node indegree and outdegree is constant**

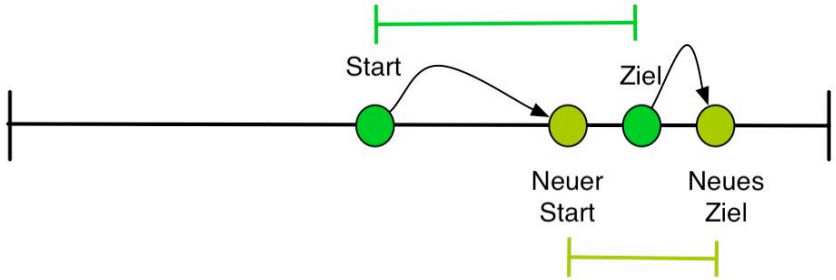




Lookup in Distance-Halving

- **Recursively perform left (or right) jump for start and target**
  - until the new start and target are the same interval
- **Lemma**
  - Lookup needs O(log n) hops and messages





Congestion during Lookup

- **Using Valiants routing trick on the hyper cube the congestion can be bounded by O(log n)**
  - if every peer has a demand of one packet
- **Idea:**
  - use random middle point for routing
  - use randomly left or right pointers to route to middle point
  - use randomly left or right pointers to route to the target point

Inserting Peers in Distance Halving

- **Theorem**
  - Inserting a peer needs O(log2 n) operations.
- **Algorithm**
  - Use multipe choice for intervals:
    - O(log n) with routing cost of O(log n) each
  - Update left and right edges

Peer-to-Peer Networks

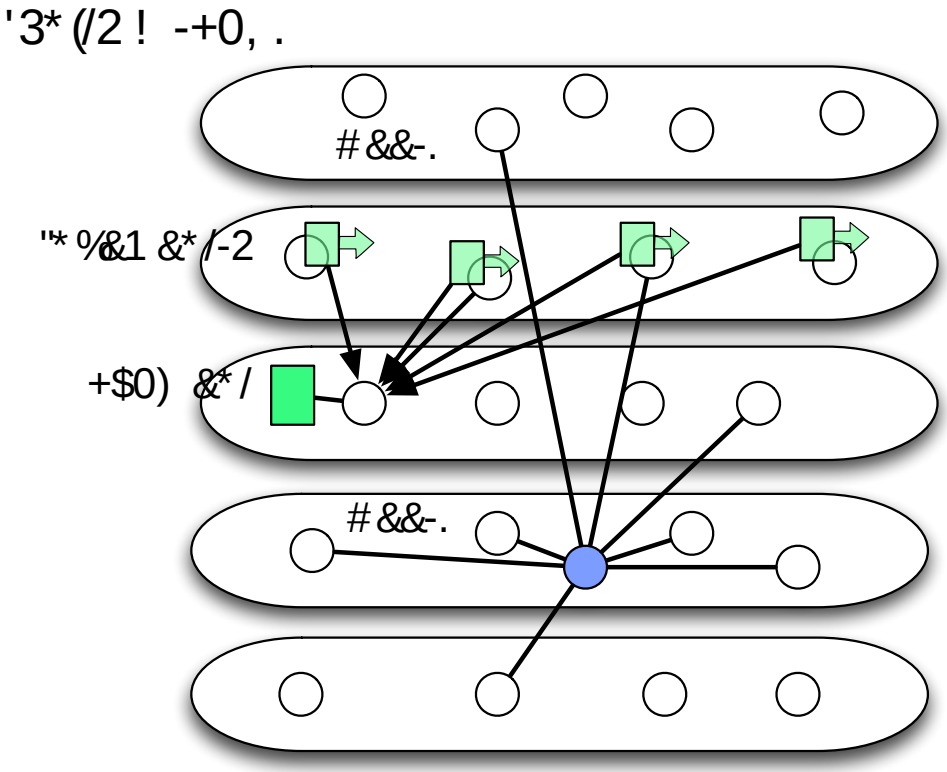
Kelips

Kelips

- **Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, Robbert van Renesse**
  - Cornell University, Ithaca, New York
- **Kelip-kelip**
  - malay name for synchronizing fireflies
- **P2P Network**
  - uses DHT
  - constant lookup time
  - $O(n^{1/2})$ storage size
  - fast and robust update

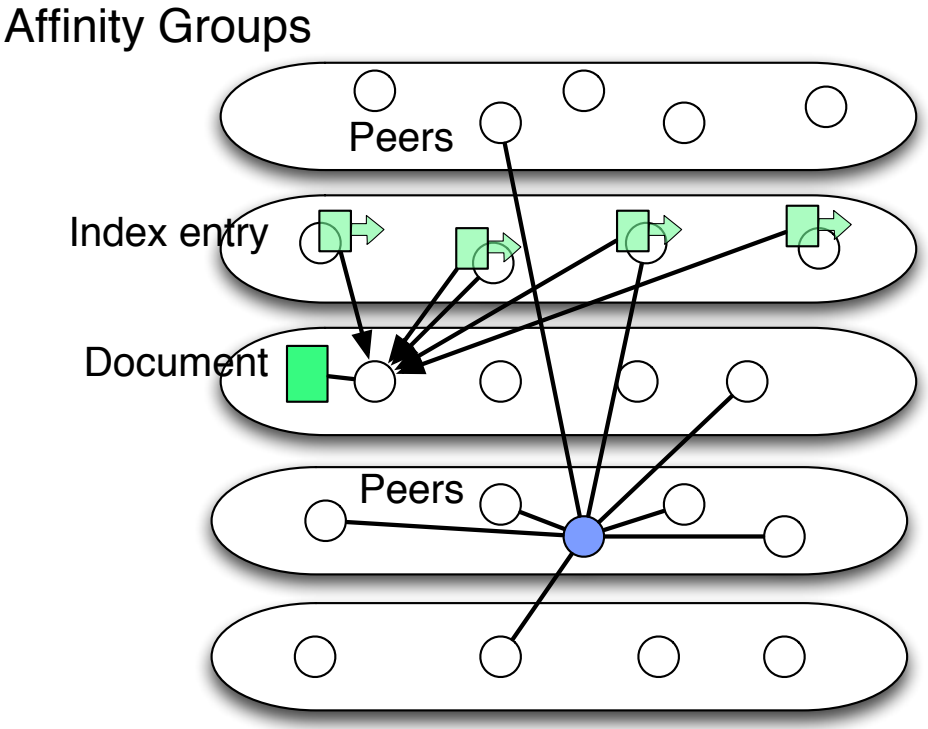
Kelips Overview

- **Peers are organized in k affinity groups**
  - peer position chosen by DHT mechanism
  - k is chosen as $n^{1/2}$ for n peers
- **Data is mapped to an affinity group using DHT**
  - all members of an affinity group store all data
- **Routing Table**
  - each peer knows all members of the affinity group
  - each peer knows at least one member of each affinity group
- **Updates**
  - are performed by epidemic algorithms



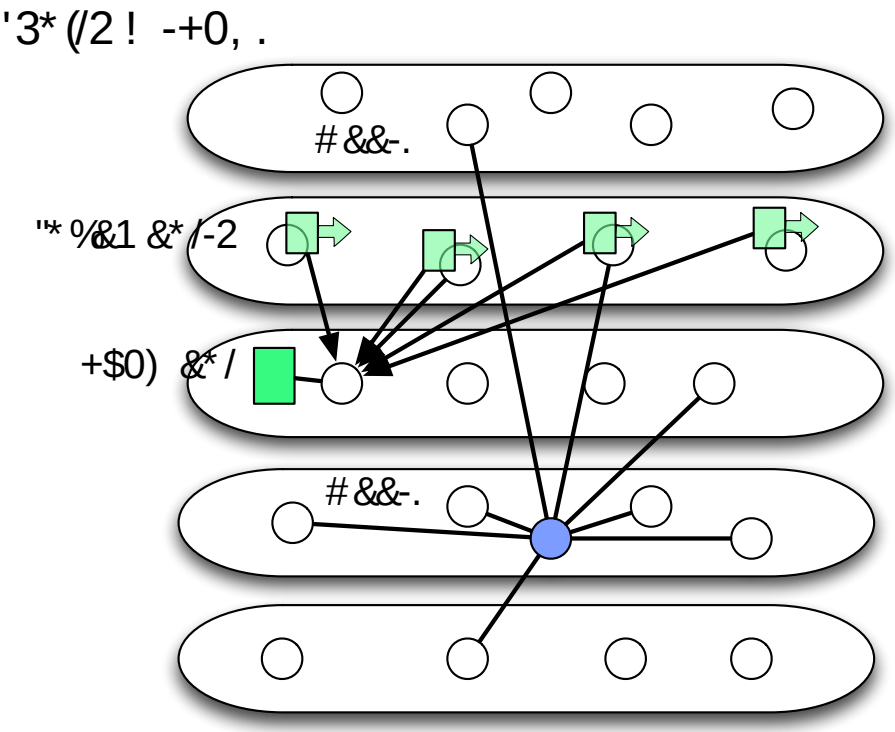
Routing Table

- **Affinity Group View**
  - Links to all O(n/k) group members
  - This set can be reduced to a partial set as long as the update mechanism works
- **Contacts**
  - For each of the other affinity group a small (constant-sized) set of nodes
  - O(k) links
- **Filetuples**
  - A (partial) set of tuples, each detailing a file name and host IP address of the node storing the file
  - O(F/k) entries, if F is the overall number of files
- **Memory Usage: O(n/k + k + F/k)**
  - for $\mathrm{k} = \mathrm{O}\left(\sqrt{n+F}\right)$ $\mathrm{O}\left(\sqrt{n+F}\right)$



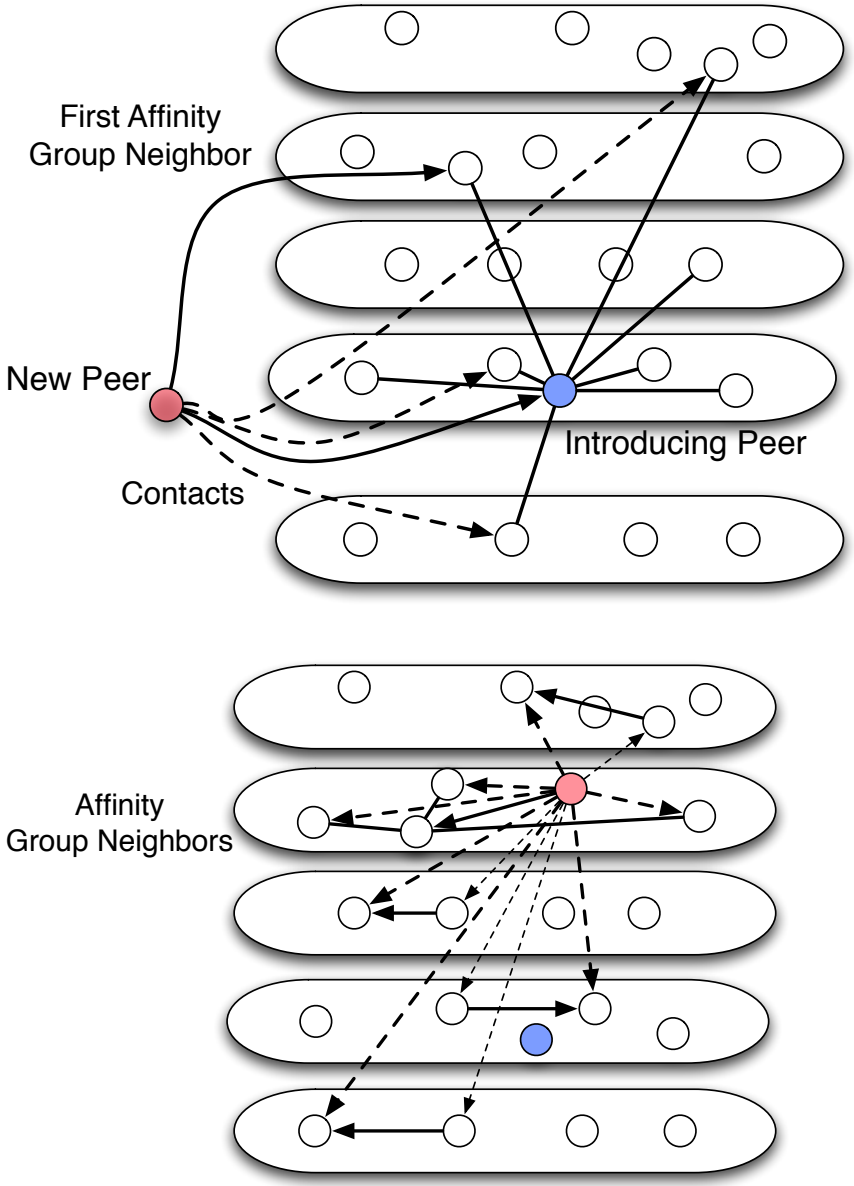
Lookup

- **Lookup-Algorithm**
  - compute index value
  - find affinity group using hash function
  - contact peer from affinity group
  - receive index entry for file (if it exists)
  - contact peer with the document
- **Kelips needs four hops to retrieve a file**



Inserting a Peer

- **Algorithm**
  - Every new peer is introduced by a special peer, group or other method,
    - e.g. web-page, forum etc.
  - The new peer computes its affinity group and contacts any peer
  - The new peer asks for one contact of the affinity group and copies the contacts of the old affinity group
  - By contacting a neighbor node in the affinity group it receives all the necessary contacts and index filetuples
  - Every contact is replaced by a random replacement (suggested by the contact peer)
  - The peer starts an **epidemic algorithm** to update all links
- **Except the epidemic algorithm the runtime is O(k) and only O(k) messages are exchanged**



How to Add a Document

- Start an Epidemic Algorithm to Spread the news in the affinity group
- Such an algorithm uses O(n/k) messages and needs O(log n) time
- We introduce Epidemic Algorithms later on

How to Check Errors

- **Kelip works in heartbeats, i.e. discrete timing**
- **In every heartbeat each peer checks one neighbor**
- **If a neighbor does not answer for some time**
  - it is declared to be dead
  - this information is spread by an epidemic algorithm
- **Using the heartbeat mechanisms all nodes also refresh their neighbors**
- **Kelips quickly detects missing nodes and updates this information**

Discussion

- **Kelips has lookup time O(1), but needs O($n^{1/2}$) sized Routing Table**
  - not counting the O($F/n^{1/2}$) Filetuples
- **Chord, Pastry & Tapestry use lookup time O(log n) but only O(log n) memory units**
- **Kelips is a reasonable choice for medium sized networks**
  - up to some million peers and some hundred thousands index entries

Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Self-Organization

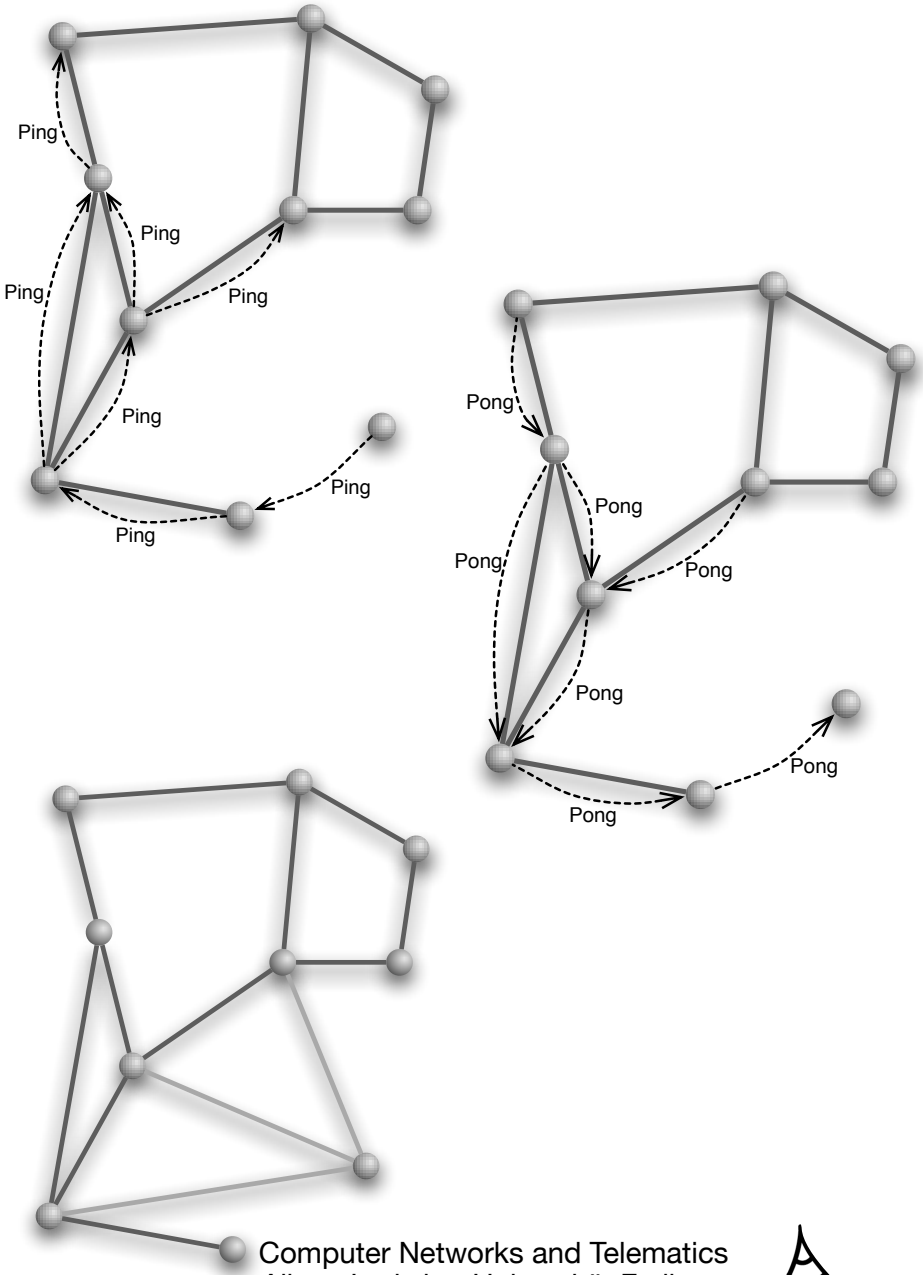
- **Chaotic networks**
  - Gnutella
  - Freenet
  - Kazaa
- **Utilizing Self-Organization**
  - Epidemic algorithms
  - Self-repairing random networks

Peer-to-Peer Networks

Gnutella

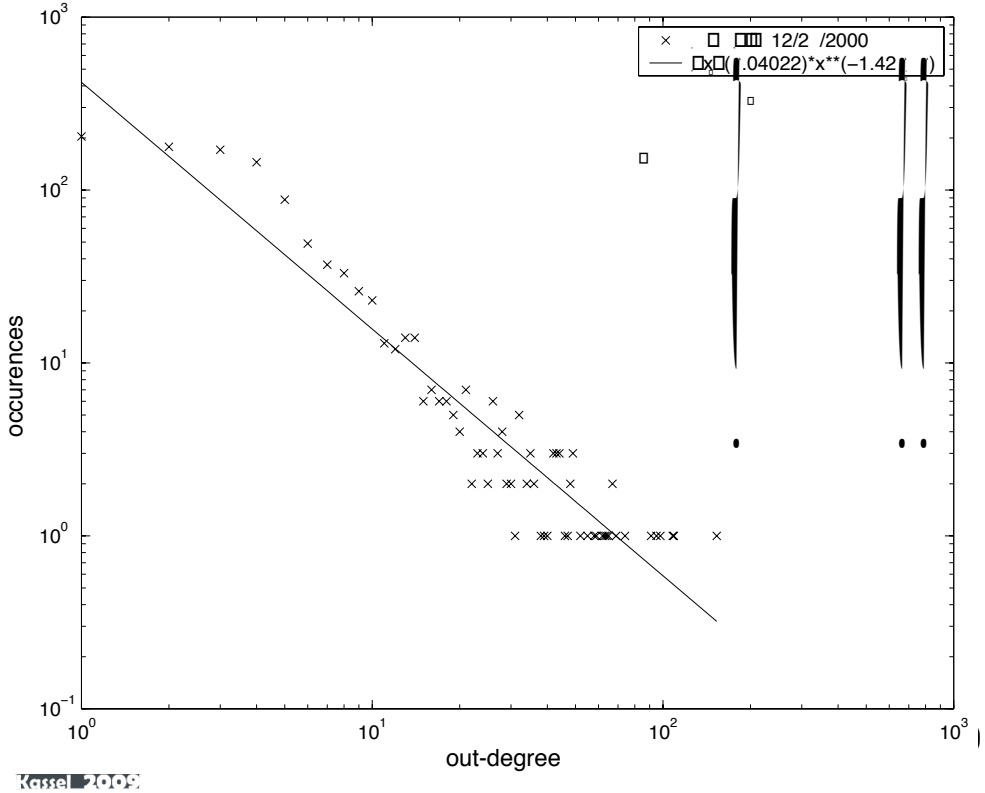
Gnutella — Connecting

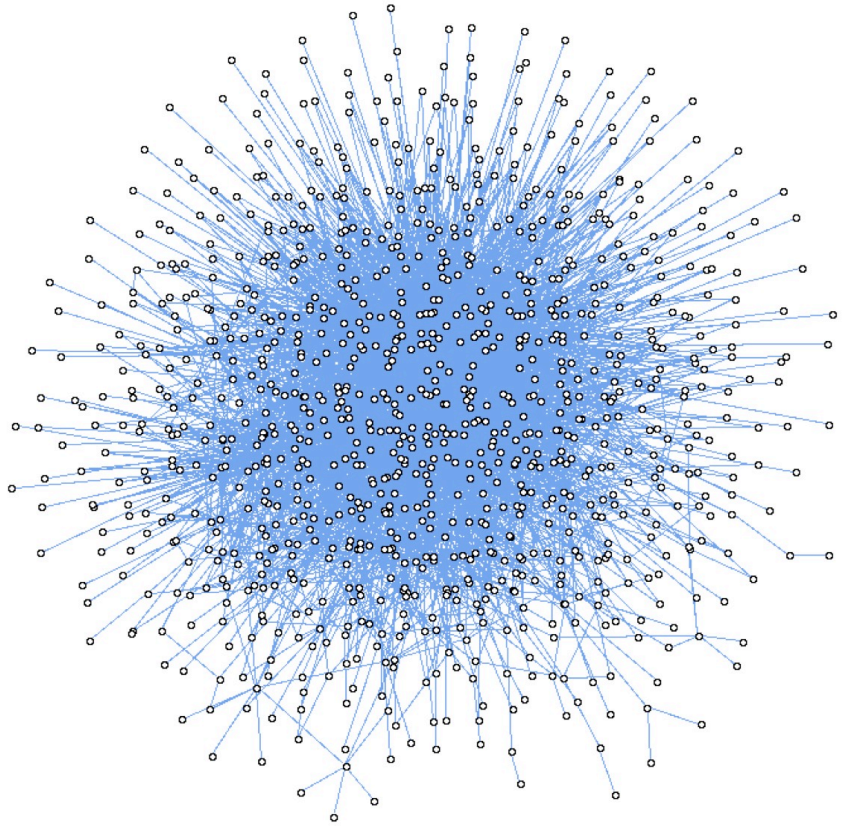
- **Protokoll**
  - Ping
    - participants query for neighbors
    - are forwarded according for TTL steps (time to live)
  - Pong
    - answers Ping
    - is forwarded backward on the query path
    - reports IP and port adress (socket pair)
    - number and size of available files



Gnutella — Graph Structure

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





Gnutella snapshot in 2000

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

CoNe Freiburg

Peer-to-Peer Networks

FreeNet 2000

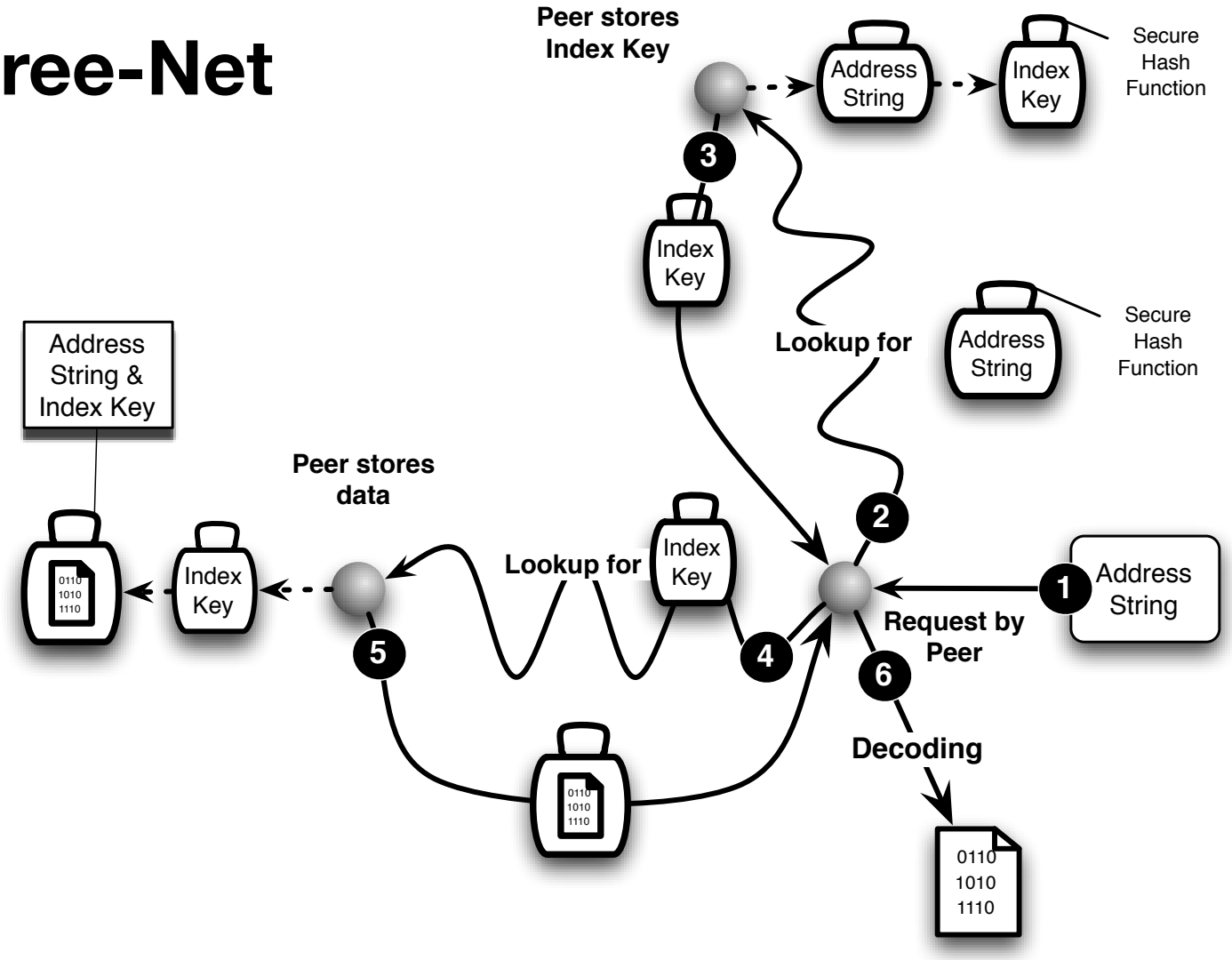
Free-Net

- **Ian Clarke, Oskar Sandberg, Brandon Wiley, Theodore Hong, 2000**
- **Goal**
  - peer-to-peer network
  - allows publication, replication, data lookup
  - anonymity of authors and readers
- **Files**
  - are encoding location independent
    - by encrypted and pseudonymously signed index files
    - author cannot be identified
  - are secured against unauthorized change or deletion
  - are encoded by keys unknown by the storage peer
    - secret keys are stored elsewhere
  - are replicated
    - on the look up path
  - and erased using “Least Recently Used” (LRU) principle

Free-Net

- **Network Structure**
  - is similar to Gnutella
  - Free-Net is like Gnutella Pareto distributed
- **Storing Files**
  - Each file can be found, decoded and read using the encoded address string and the signed subspace key
  - Each file is stored together with the information of the index key but without the encoded address string
  - The storage peer cannot read his files
    - unless he tries out all possible keywords (dictionary attack)
- **Storing of index files**
  - The address string coded by a cryptographic secure hash function leads to the corresponding peer
    - who stores the index data
      - address string
      - and signed subspace key
  - Using this index file the original file can be found

Free-Net


Peer stores
Index Key
Address
String
Index
Key
Secure
Hash
Function
3
Index
Key
Lookup for
Address
String
Secure
Hash
Function
Address
String &
Index Key
Peer stores
data
2
Index
Key
Lookup for
Index
Key
1
Address
String
Request by
Peer
5
4
6
Decoding
0110
1010
1110

Free-Net

- **Lookup**
  - steepest-ascent hill-climbing
    - lookup is forwarded to the peer whose ID is closest to the search index
  - with TTL field
    - i.e. hop limit
- **Files are moved to new peers**
  - when the keyword of the file is similar to the neighbor‘s ID
- **New links**
  - are created if during a lookup close similarities between peer IDs are discovered

Efficiency of Free-Net

- **Network structure of Free-Net is similar to Gnutella**

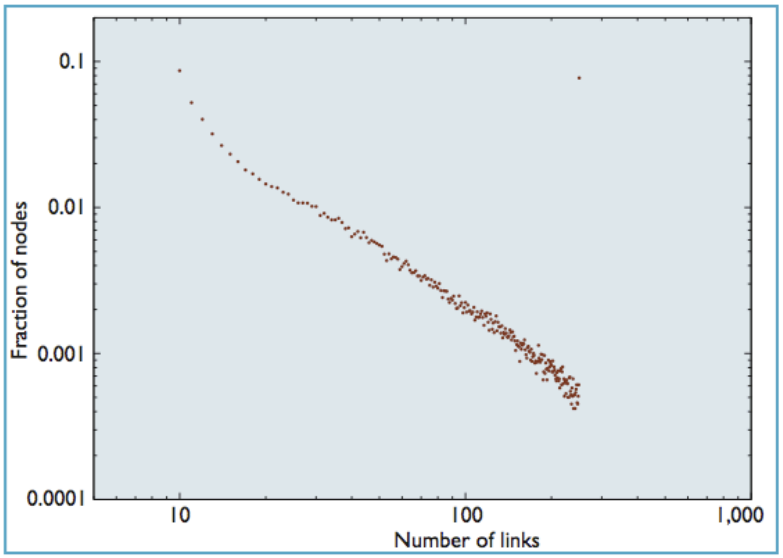


Figure 2. Degree distribution among Freenet nodes.The network shows a close fit to a power-law distribution.

- **The lookup time is polynomial on the average**

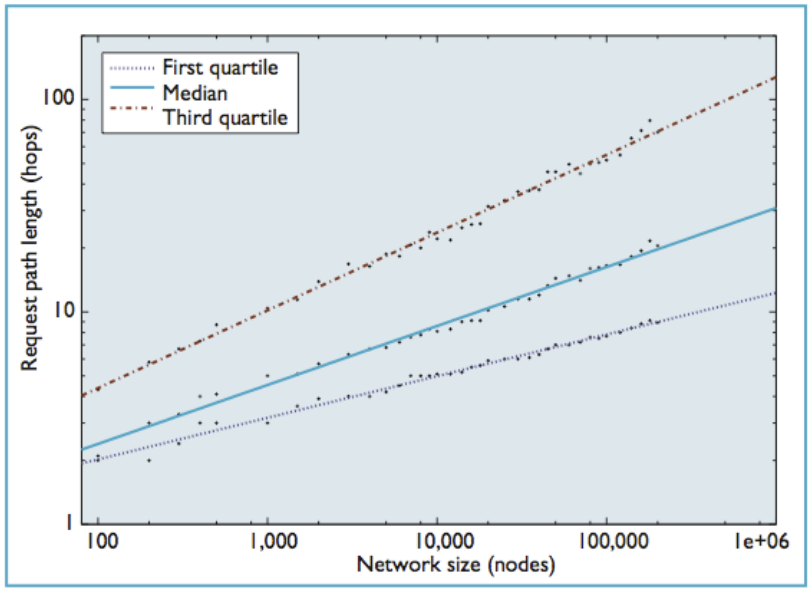


Figure 3. Request path length versus network size.The median path length in the network scales as $N^{0.28}$.

Peer-to-Peer Networks

Epidemic Algorithms

Epidemic Spread of Viruses

- **Observation**
  - most viruses do not prosper in real life
  - other viruses are very successful and spread fast
- **How fast do viruses spread?**
- **How many individuals of the popolation are infected?**
- **Problem**
  - social behavior and infection risk determine the spread
  - the reaction of a society to a virus changes the epidemy
  - viruses and individuals may change during the infection

Mathematical Models

- **SI-Model (rumor spreading)**
  - susceptible → infected
- **SIS-Model (birthrate/deathrate)**
  - susceptible → infected → susceptible
- **SIR-Model**
  - susceptible → infected → recovered

- **Continuos models**
  - deterministic
  - or stochastic
- **Lead to differential equations**

- **Discrete Models**
  - graph based models
  - random call based
- **Lead to the analysis of Markov Processes**

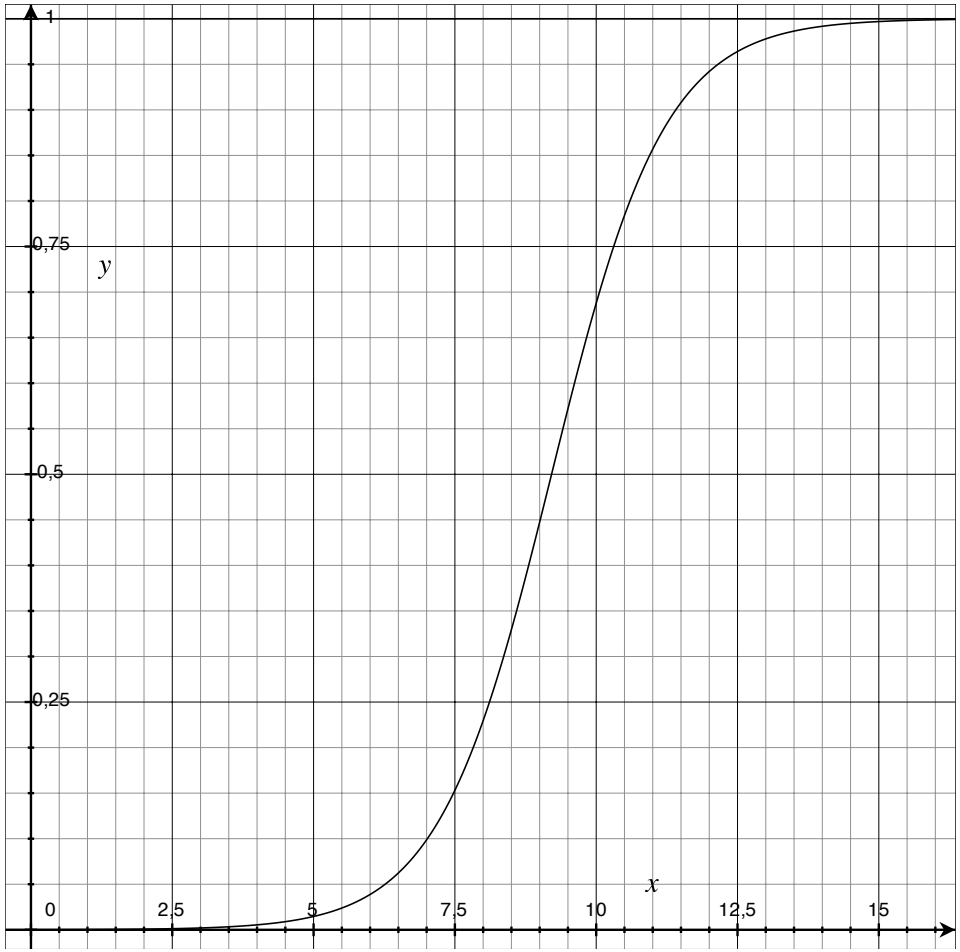
Infection Models

- **SI-Model (rumor spreading)**
  - susceptible → infected
  - At the beginning one individual is infected
  - Every contact infects another indiviual
  - In every time unit there are in the expectation ß contacts
- **SIS-Model (birthrate/deathrate)**
  - susceptible → infected → susceptible
  - similar as in the SI-Model, yet a share of δ of all infected individuals is healed and can receive the virus again
  - with probability δ an individual is susceptible again
- **SIR-Model**
  - susceptible → infected → recovered
  - like SI-Model, but healed individuals remain immune against the virus and do not transmit the virus again

SI-Model

- The number of infected grows exponentially until half of all members are infected
- Then the number of susceptible decrease exponentially

$$i(t) = \frac{1-\rho}{1+\left(\frac{1-\rho}{i(0)}-1\right)e^{-(\beta-\delta)t}}$$



Replicated Databases

- **Same data storage at all locations**
  - new entries appear locally
- **Data must be kept consistently**
- **Algorithm is supposed to be decentral and robust**
  - since connections and hosts are unreliable
- **Not all databases are known to all**
- **Solutions**
  - Unicast
    - New information is sent to all data servers
  - Problem:
    - not all data servers are known and can be reached
  - Anti-Entropy
    - Every local data server contacts another one and exchanges all information
    - total consistency check of all data
  - Problem
    - comunication overhead
- **Epicast …**

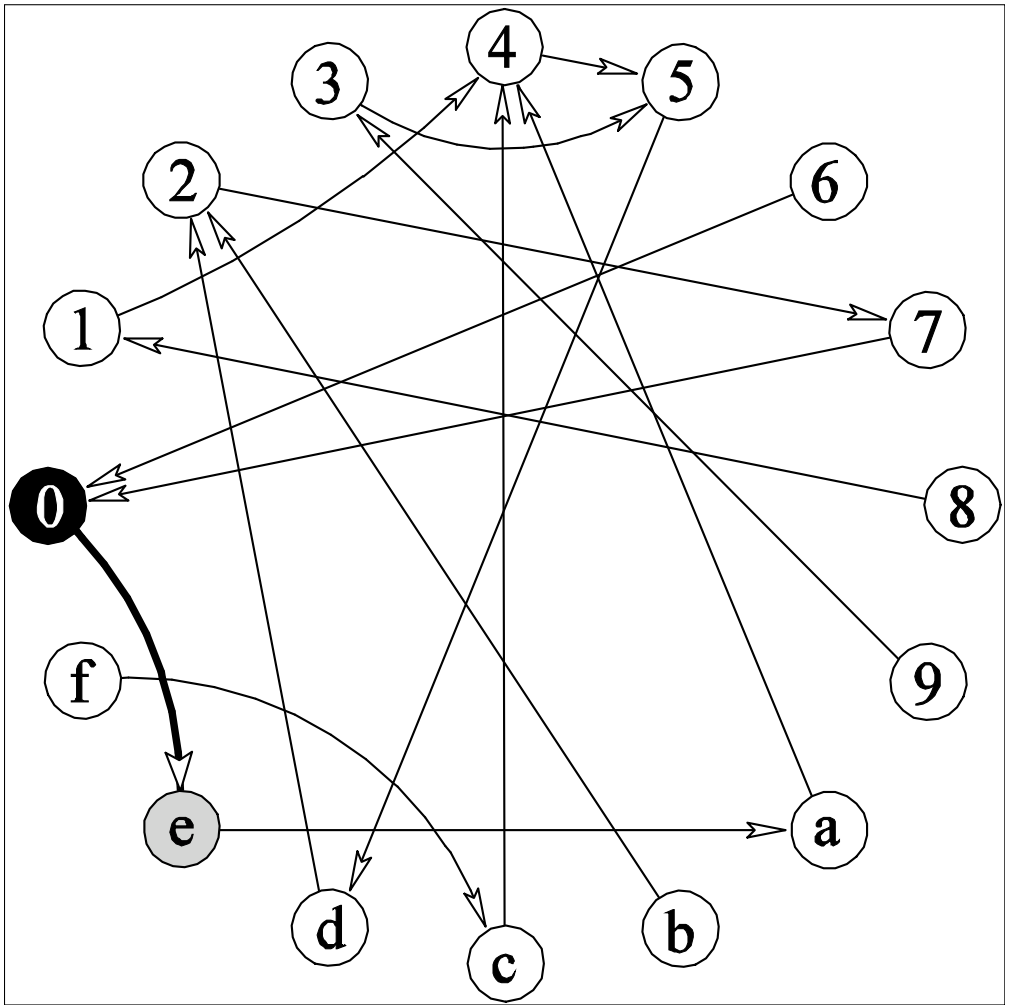
Epidemic Algorithms

- **Epicast**
  - new information is a rumor
  - as long the rumor is new it is distributed
  - Is the rumor old, it is known to all servers
- **Epidemic Algorithm [Demers et al 87]**
  - distributes information like a virus
  - robust alternative to BFS or flooding
- **Communication method**
  - Push & Pull, d.h. infection after $\log_3 n + O(\log \log n)$ rounds with high probability
- **Problem:**
  - growing number of infections increases comunication effort
  - trade-off between robustness and communication overhead

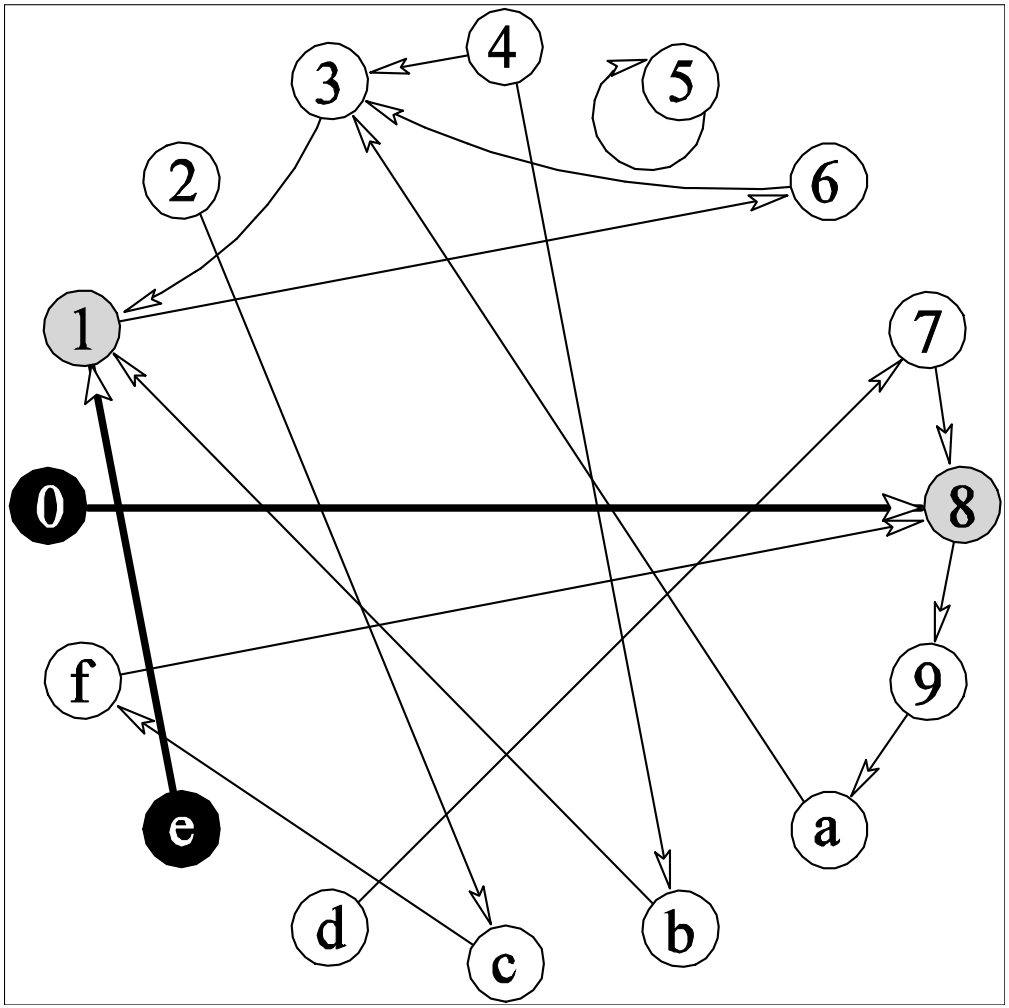
Random Call Model

- **In each round a new contact graph $G_t=(V,E_t)$:**
  - Each node in G_t has out-degree 1
    - chooses random node v out of V
- **Infection models:**
  - Push-Model
    - if u is infected and $(u,v) \in E_t$, then v is infected in the next round
  - Pull-Modell:
    - if v is infected and $(u,v) \in E_t$, then u is infected in the next round

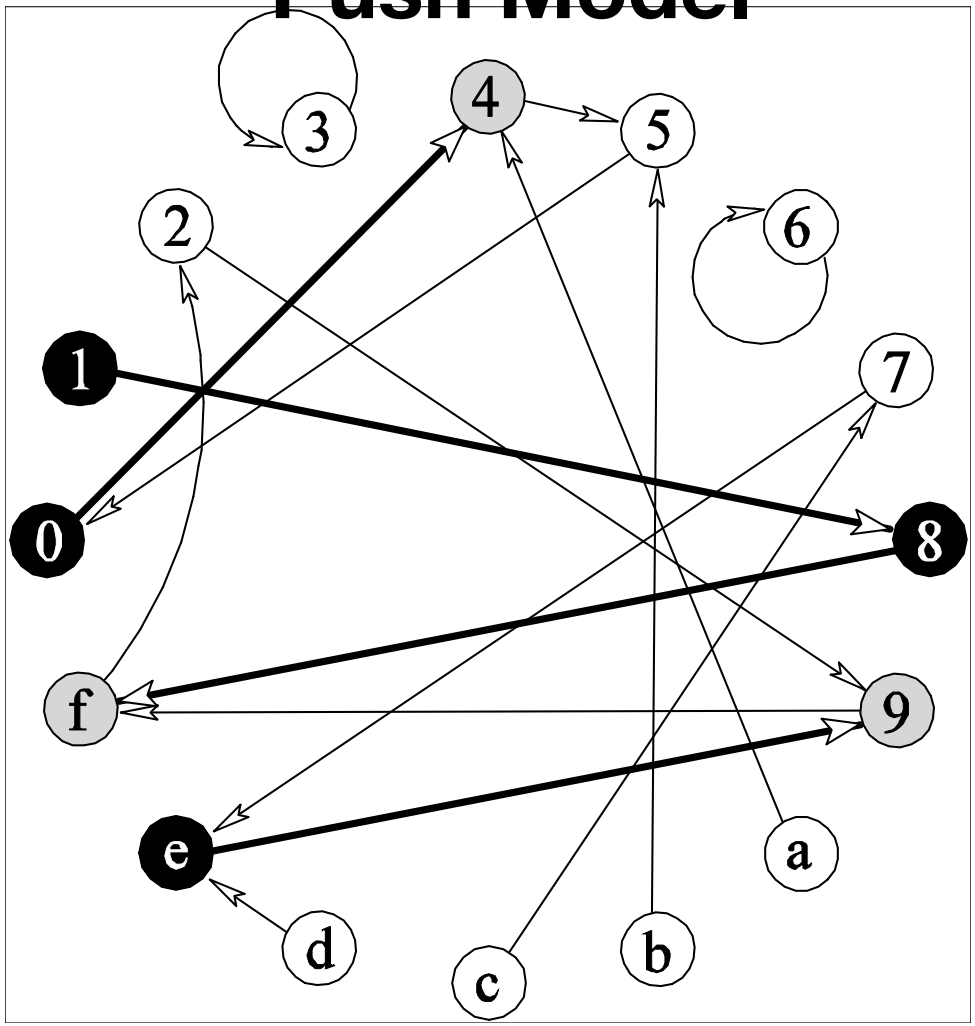
Push Model



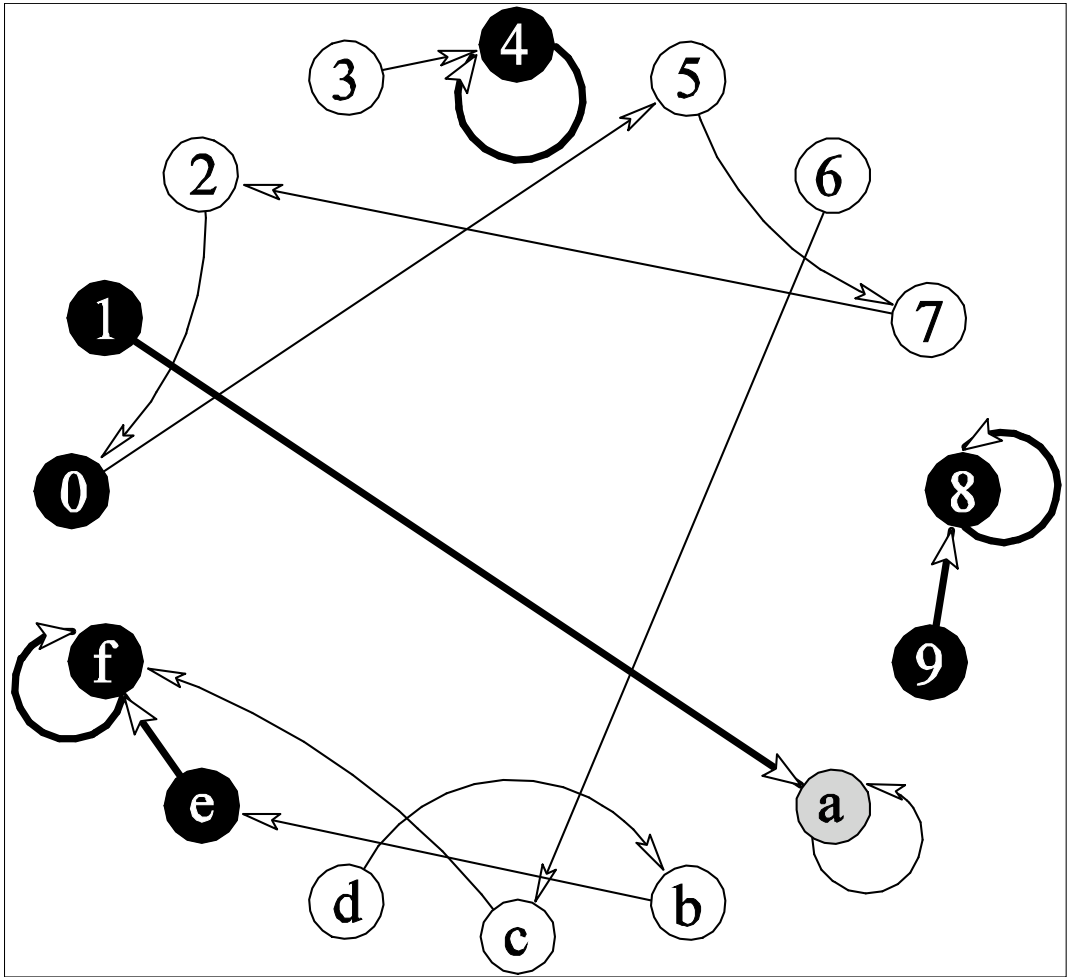
Push Model



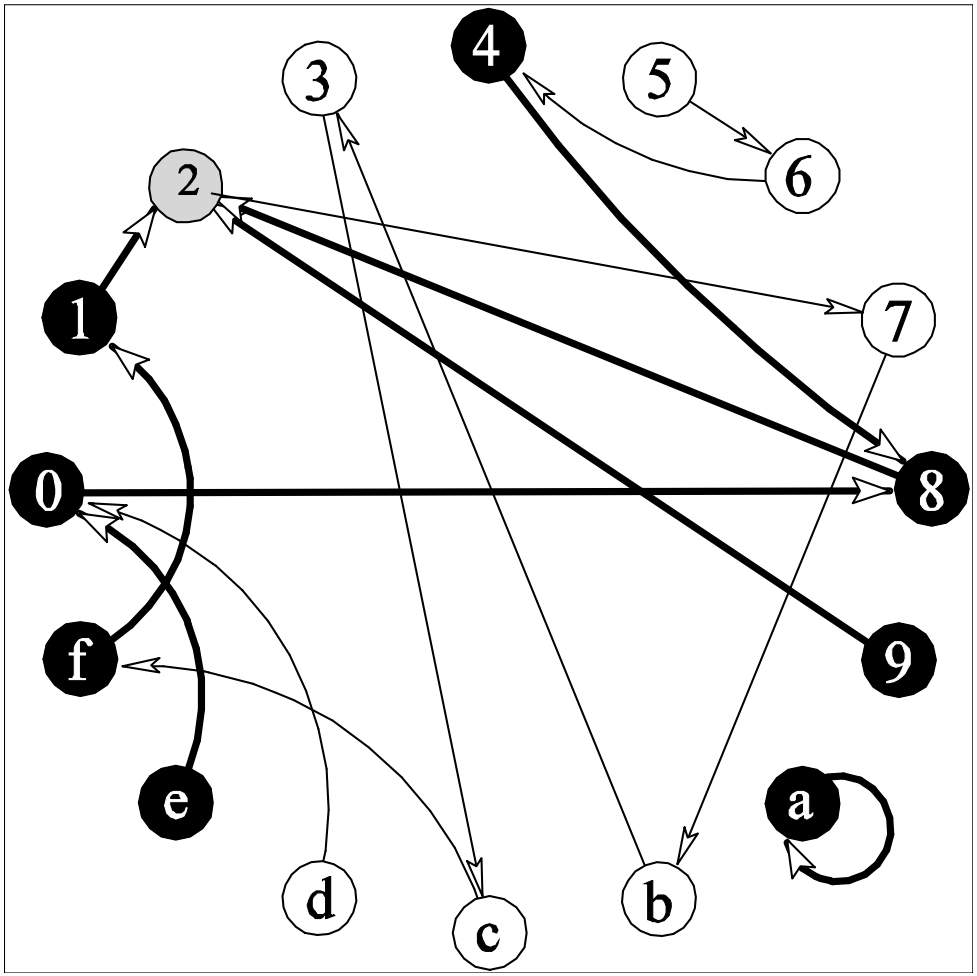
Push Model



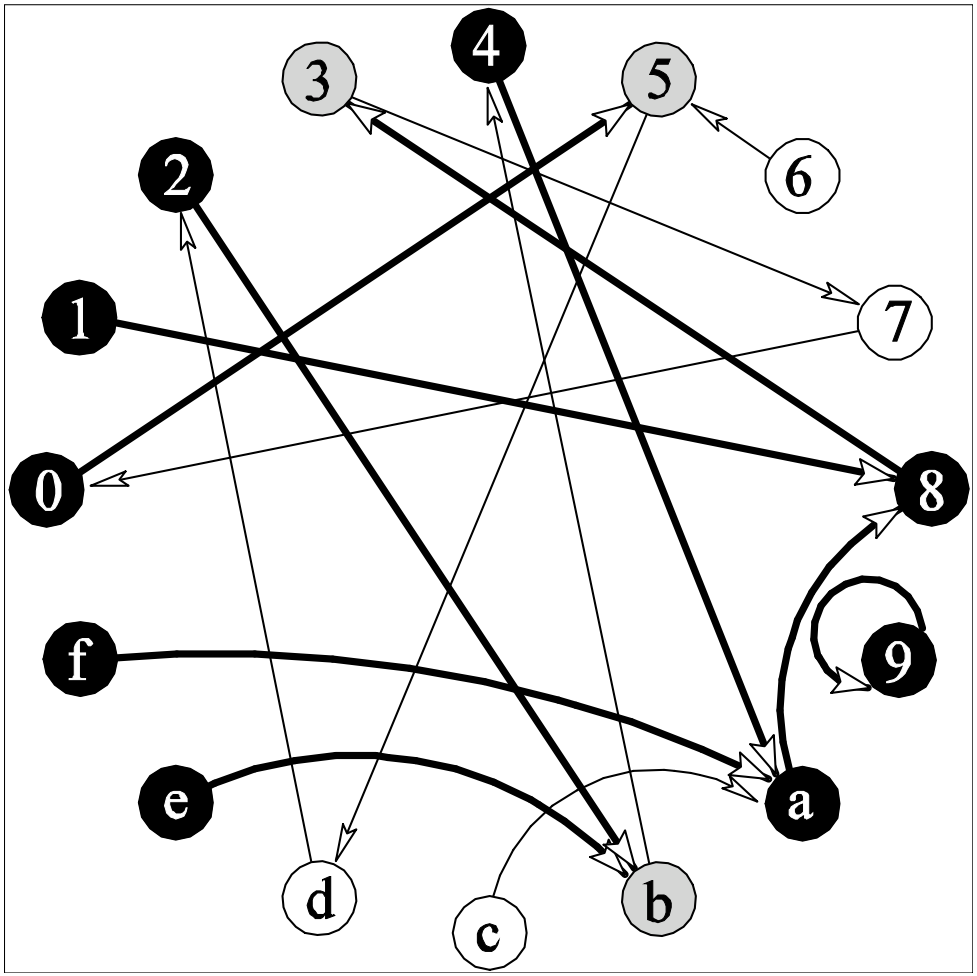
Push Model



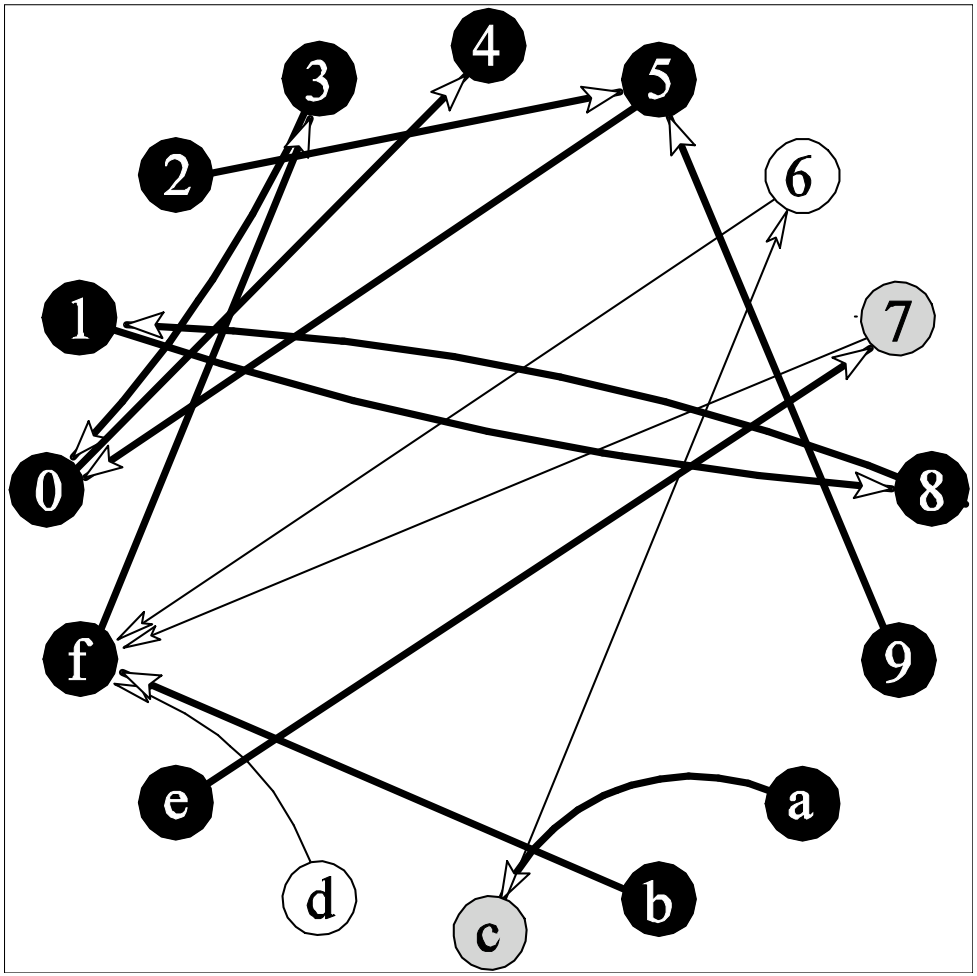
Push Model



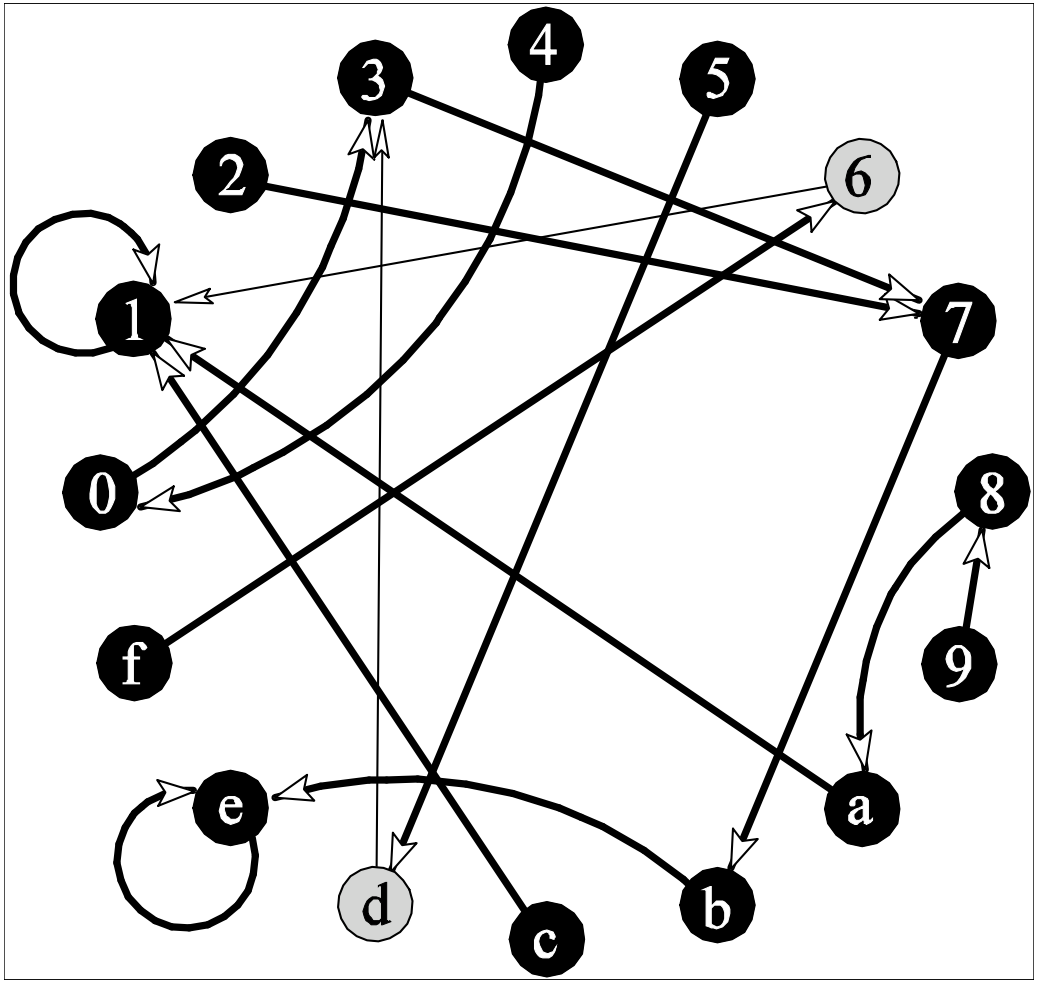
Push Model



Push Model



Push Model



Push Model
Start Phase

- **3 cases for an infected node**
  1. he is the only one infecting a new node
  2. he contacts an already infected node
  3. he infects together with other infected nodes a new node
     - this case is neglected in the prior deterministic case
  - Probability for 1st or 3rd case $s(t) = 1-i(t)$
  - Probability for 2nd case $i(t)$
  - Probability for 3rd case is at most $i(t)$
    - since at most $i(t)$ are infected
- **Probability of infection of a new node, if $i(t) \leq s(t)/2$:**
  - at least $1 - 2i(t)$
- **$E[i(t+1)] \geq i(t) + i(t)(1 - 2\,i(t)) = 2i(t) - 2i(t)^2 \approx 2\,i(t)$**

Push Model
Start phase & Exponential Growth

- **If $i(t) \leq s(t)/2$:**
  - $E[i(t+1)] \geq 2\, i(t) - 2i(t)2 \approx 2\, i(t)$
- **Start phase: $I(t) \leq 2\, c\, (\ln n)^2$**
  - Variance of i(t+1) relatively large
  - Exponential growth starts after some O(1) with high probability
- **Exponential growth:**
  $I(t) \in [2\, c\, (\ln n)^2, n/(\log n)]$
  - Nearly doubling of infecting nodes with high probability, i.e. $1\text{-}O(n^{-c})$

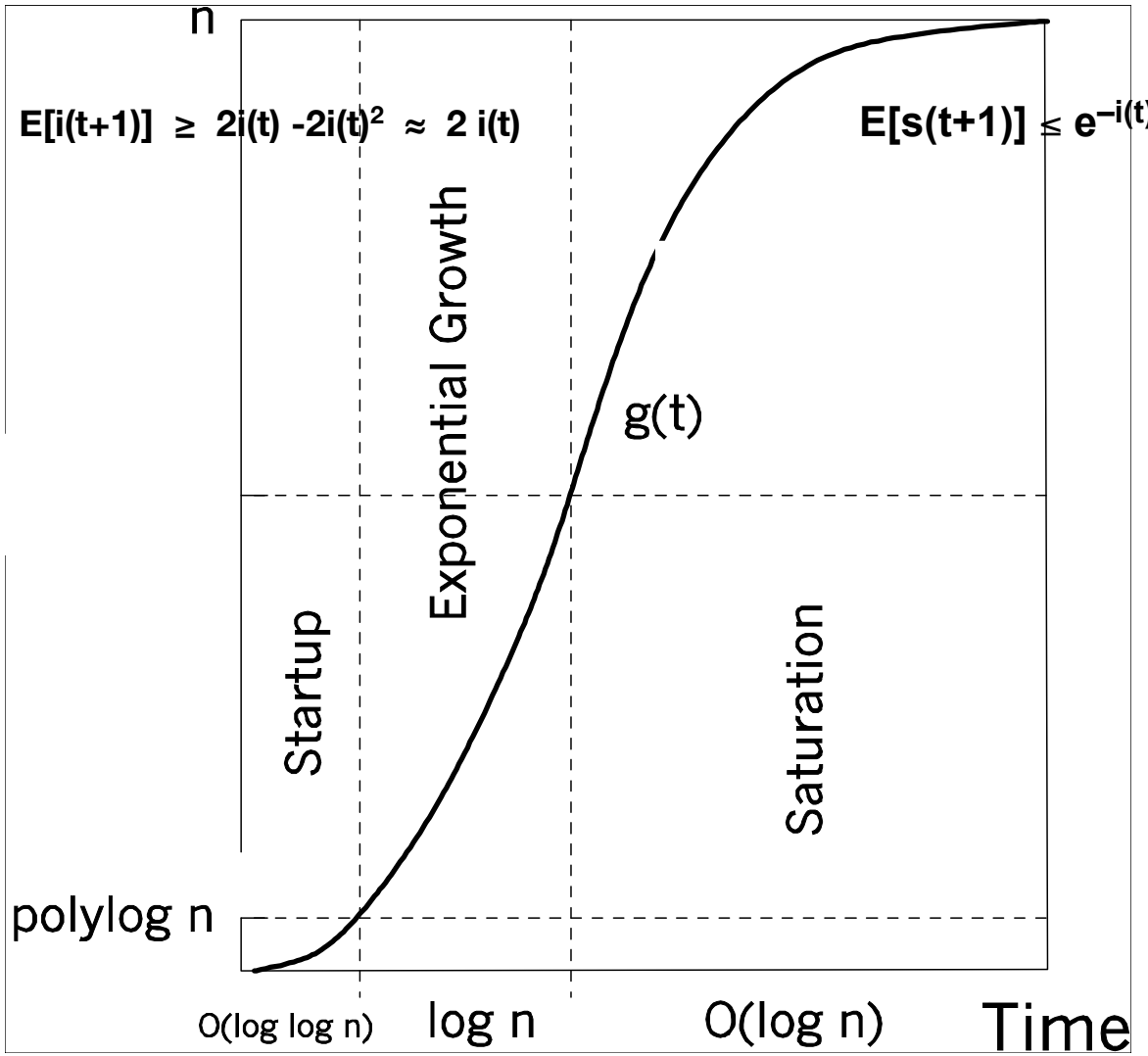
Push Model
Middle Phase & Saturation

- Probability of infections of a new node if $i(t) \leq s(t)/2$: **$1 - 2i(t)$**
  - **$E[i(t+1)] \geq 2\,i(t) - 2i(t)^2 \approx 2\,i(t)$**
- **Middle phase $I(t) \in [n/(\log n), n/3]$**
  - term **$2i(t)^2 \geq 2i(t)/(\log n)$** cannot be neglected anymore
  - Yet, **$2i(t) - 2i(t)^2 \geq 4/3\,i(t)$** still implies expontential growth, but with base < 2
- **Saturation: $I(t) \geq n/3$**
  - Probability that a susceptible node is not contacted by $I(t) = c\,n$ infected nodes:

$$\left(1 - \frac{1}{n}\right)^{cn} = \left(\left(1 - \frac{1}{n}\right)^{n}\right)^{c} \leq \frac{1}{e^c}$$

- This implies a constant probability for infection $\geq$ **$1 - e^{-1/3}$** und $\leq$ **$1 - e^{-1}$**
- Hence **$E[s(t+1)] \leq e^{-i(t)}\,s(t) \leq e^{-1/3}\,s(t)$**
- Chernoff-bounds imply that this holds with high probability
- Exponential shrinking of susceptible nodes
- Base converges to 1/e

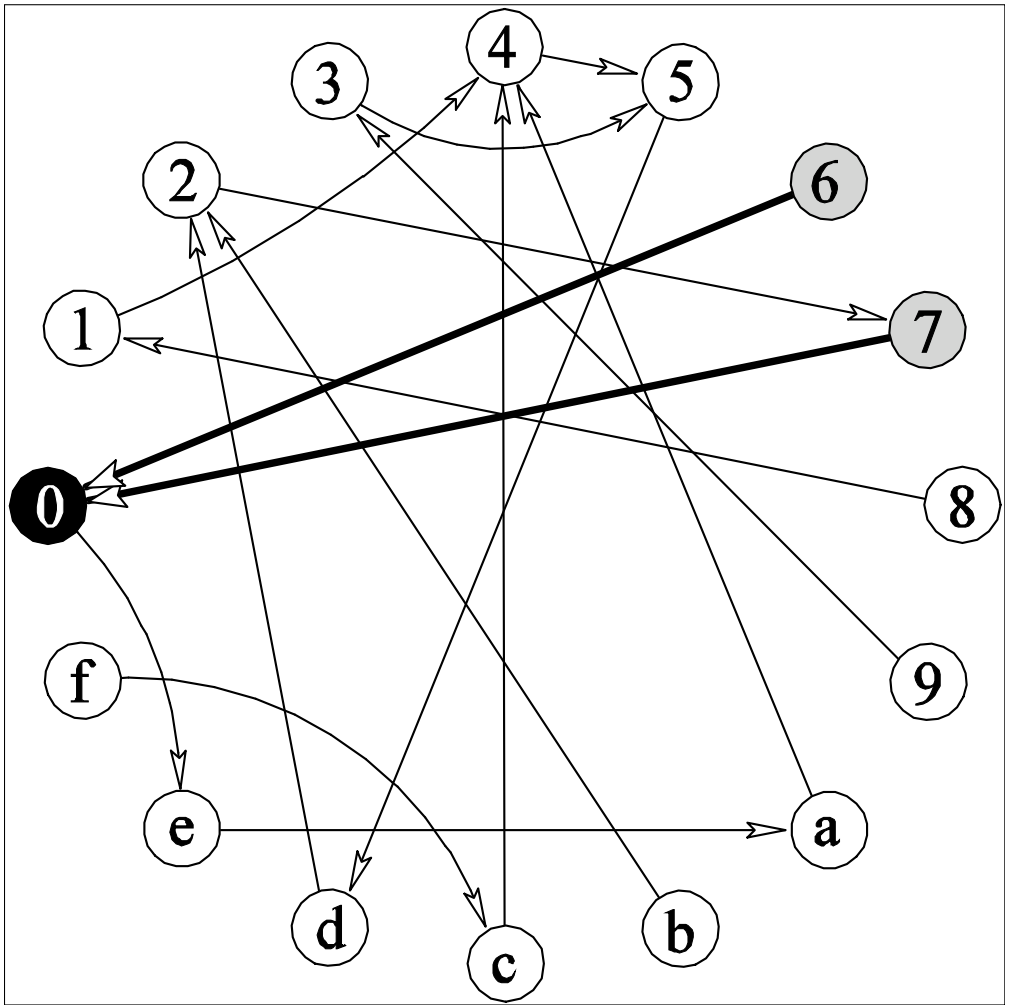
Push Model


n
E[i(t+1)] ≥ 2i(t) -2i(t)² ≈ 2 i(t)
E[s(t+1)] ≤ e^-i(t) s(t) ≤ e^-1/3 s(t)
Exponential Growth
g(t)
Startup
Saturation
polylog n
O(log log n)
log n
O(log n)
Time

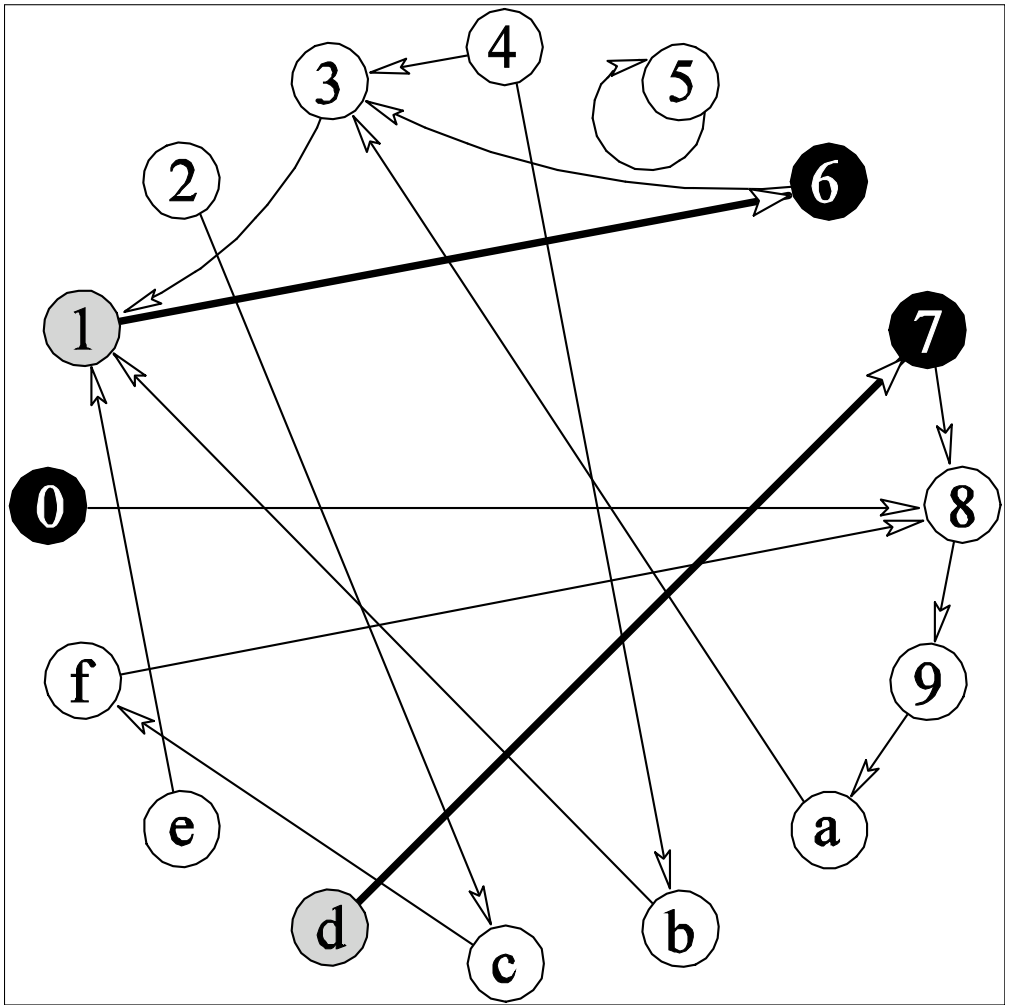
Random Call Model

- **Infection models:**
  - Push Model
    - if u is infected and $(u,v) \in E_t$, then v is infected in the next round
  - Pull Model
    - if v is infected and $(u,v) \in E_t$, then u is infected in the next round

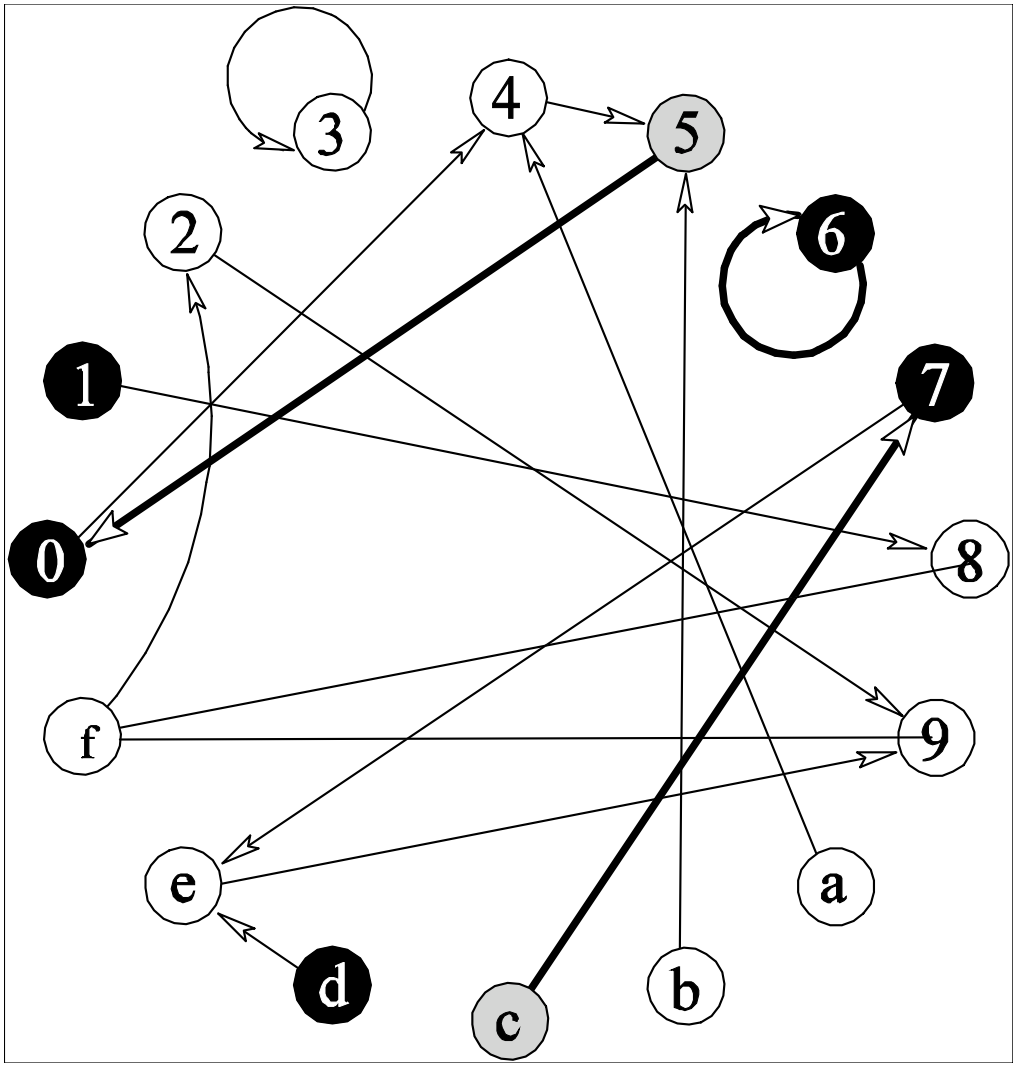
Pull Model



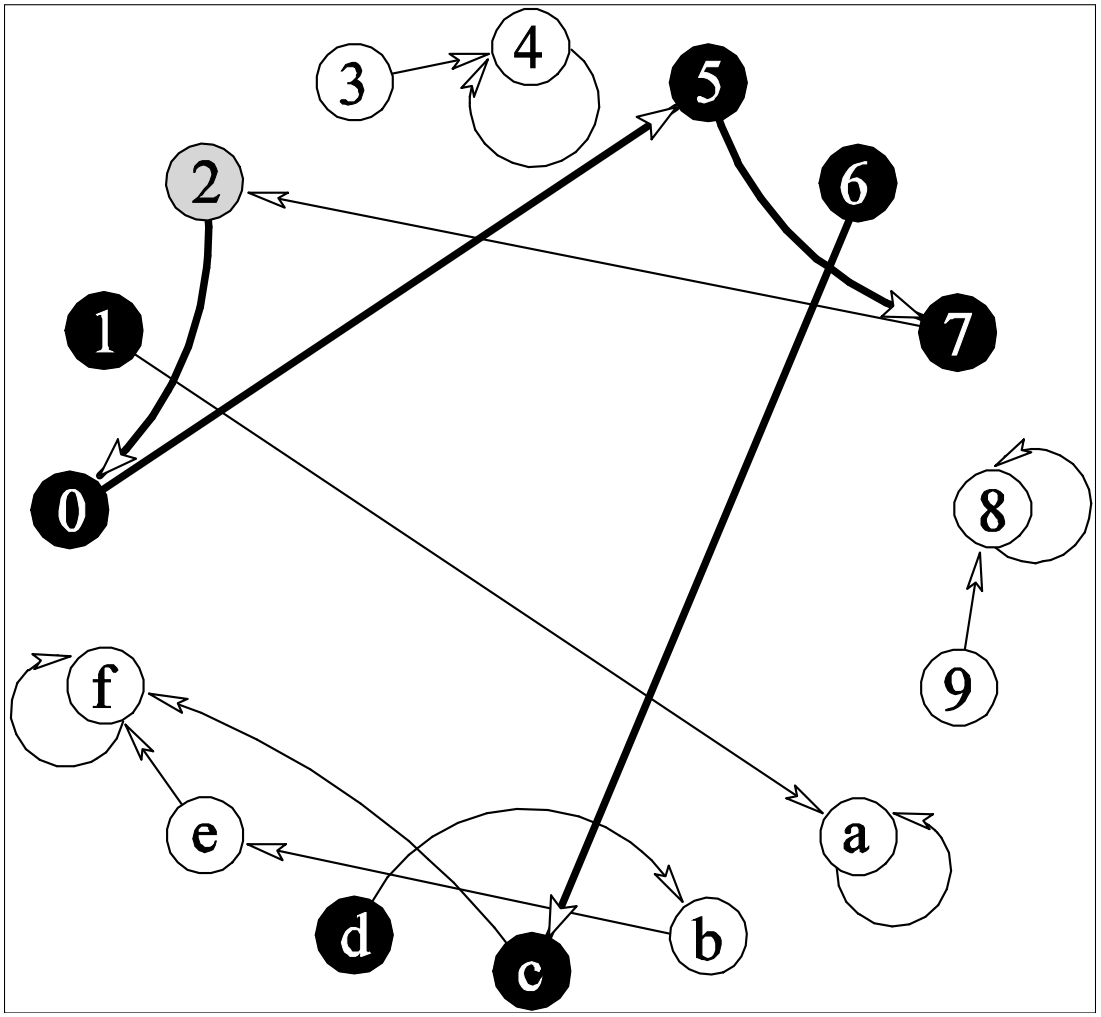
Pull Model



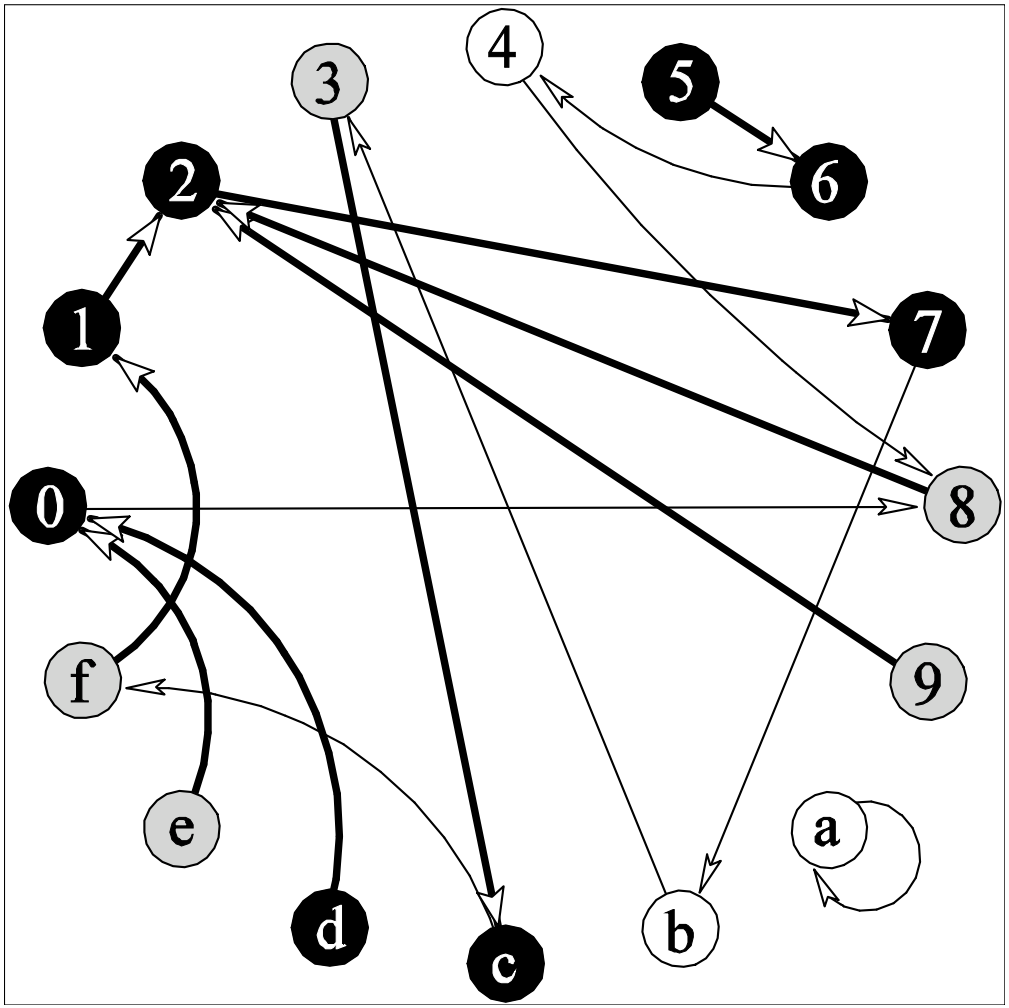
Pull Model



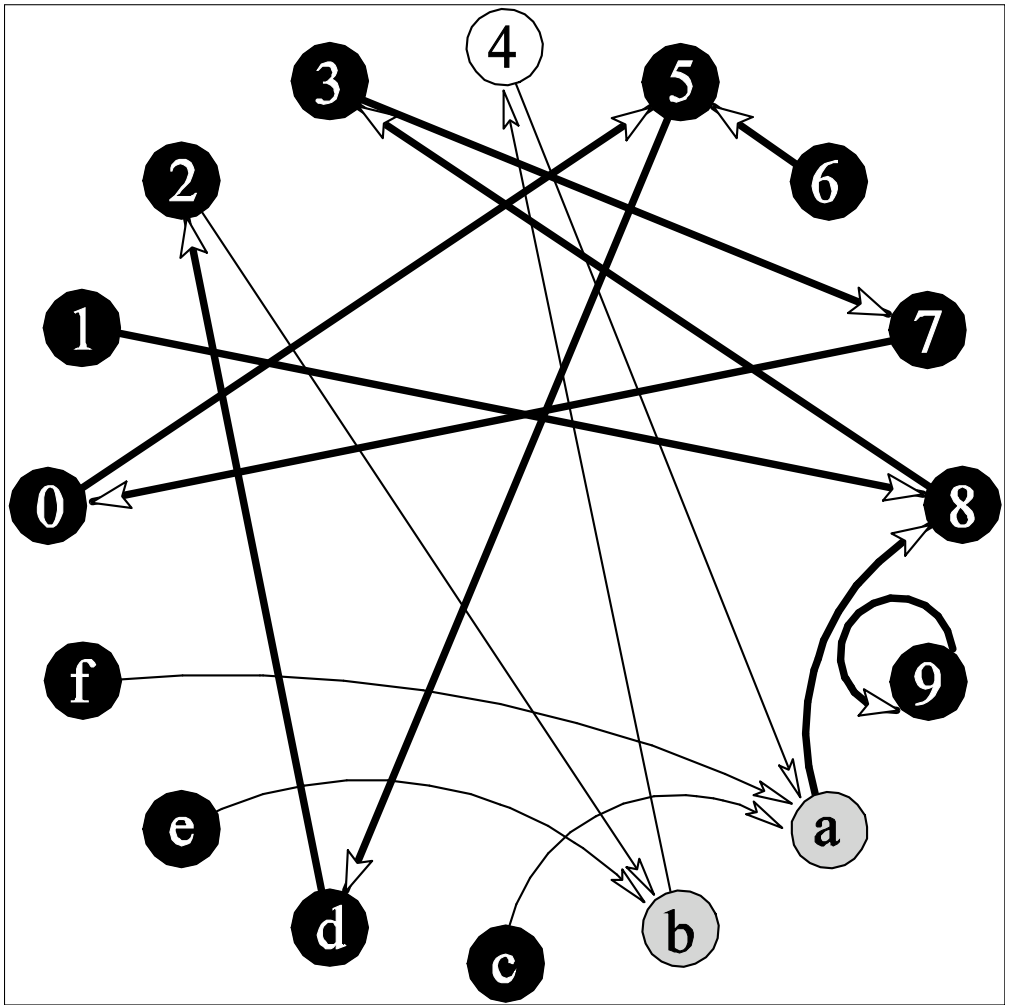
Pull Model



Pull Model



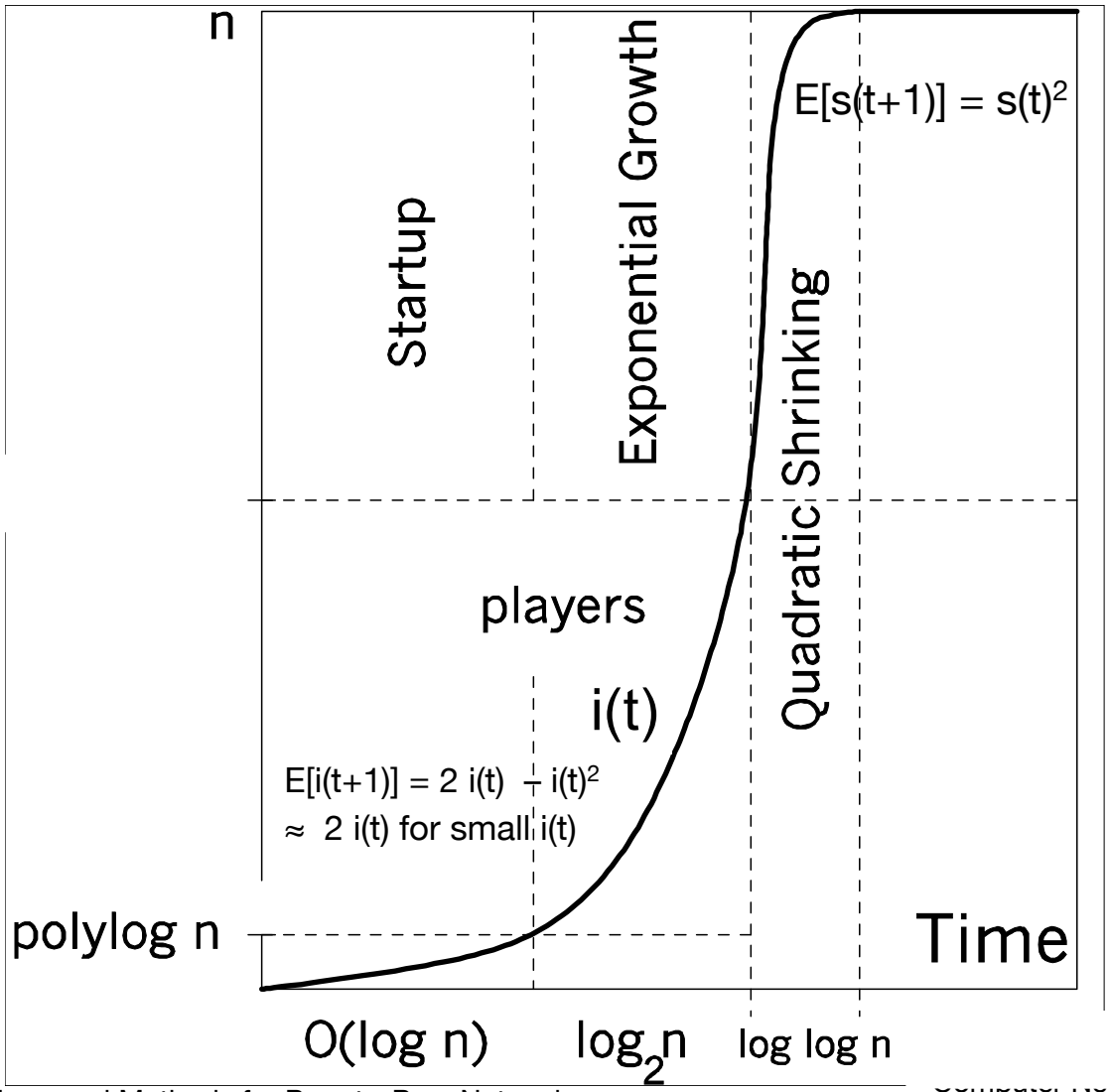
Pull Model



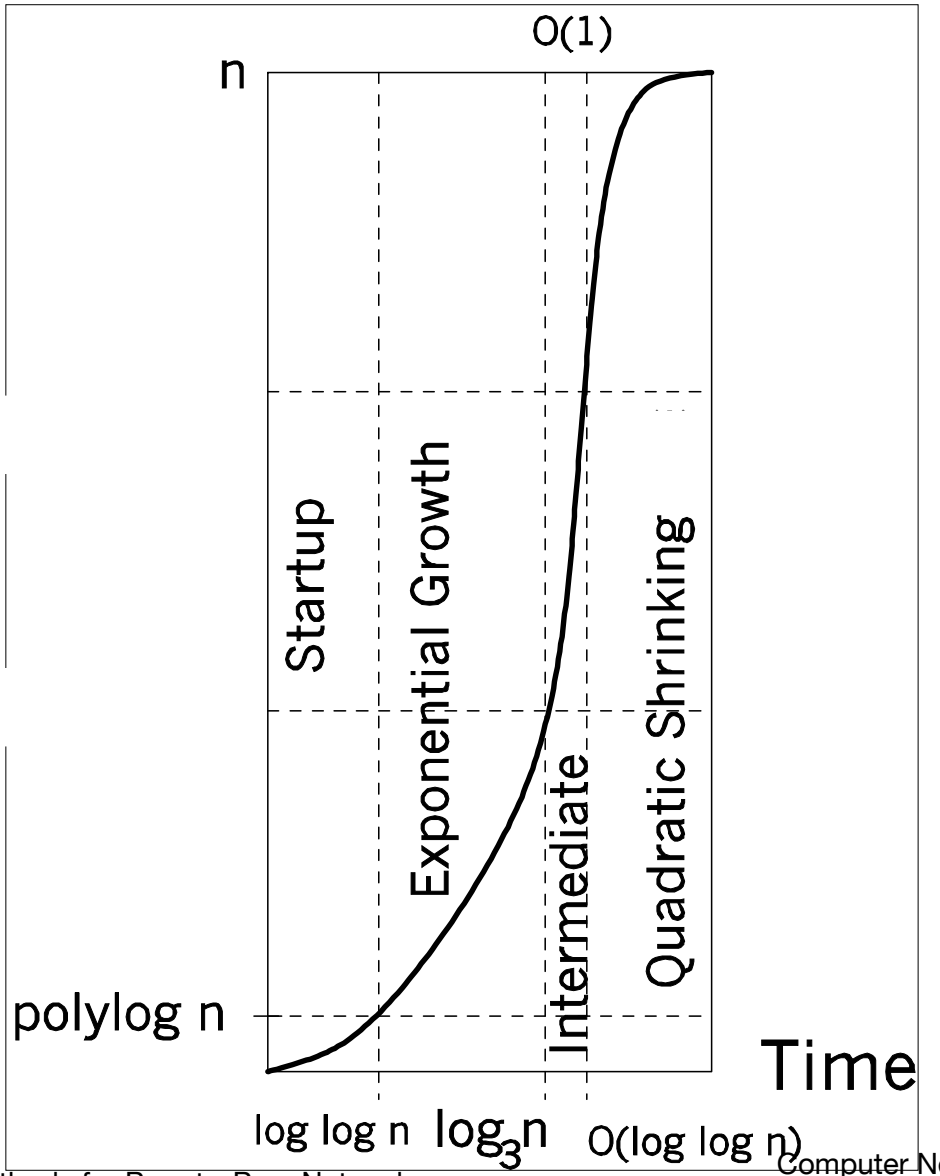
Pull Model

- **Consider**
  - an susceptible node and I(t) infected nodes
- **Probability that a susceptible node contacts an infected node: i(t)**
  - E[s(t+1)]
    = s(t) – s(t) i(t)
    = s(t) (1 – i(t)) = $s(t)^2$
  - E[i(t+1)]
    = 1-$s(t)^2$
    = 1 – $(1 – i(t))^2$
    = 2 i(t) – $i(t)^2$ ≈ 2 i(t) for small i(t)
- **Problem**
  - if i(t) ≤ $(\log n)^2$ then exponential growth is not with high probability
  - O(log n) steps are needed to start eh growth with high probability
    - yet in the expectation it grows exponentially
- **After this phase**
  - If s(t) ≤ ½
    - then the share of susceptible nodes is squared in each step
  - This implies E[s(t+ O(log log n))] = 0,
  - If i(t) ≥ ½ then after O(log log n) steps all nodes are infected with high probability

Pull Model


n
Startup
Exponential Growth
Quadratic Shrinking
E[s(t+1)] = s(t)²
players
i(t)
E[i(t+1)] = 2 i(t) – i(t)²
≈ 2 i(t) for small i(t)
polylog n
Time
O(log n)
log₂n
log log n

Push&Pull Model



Max-Counter Algorithm

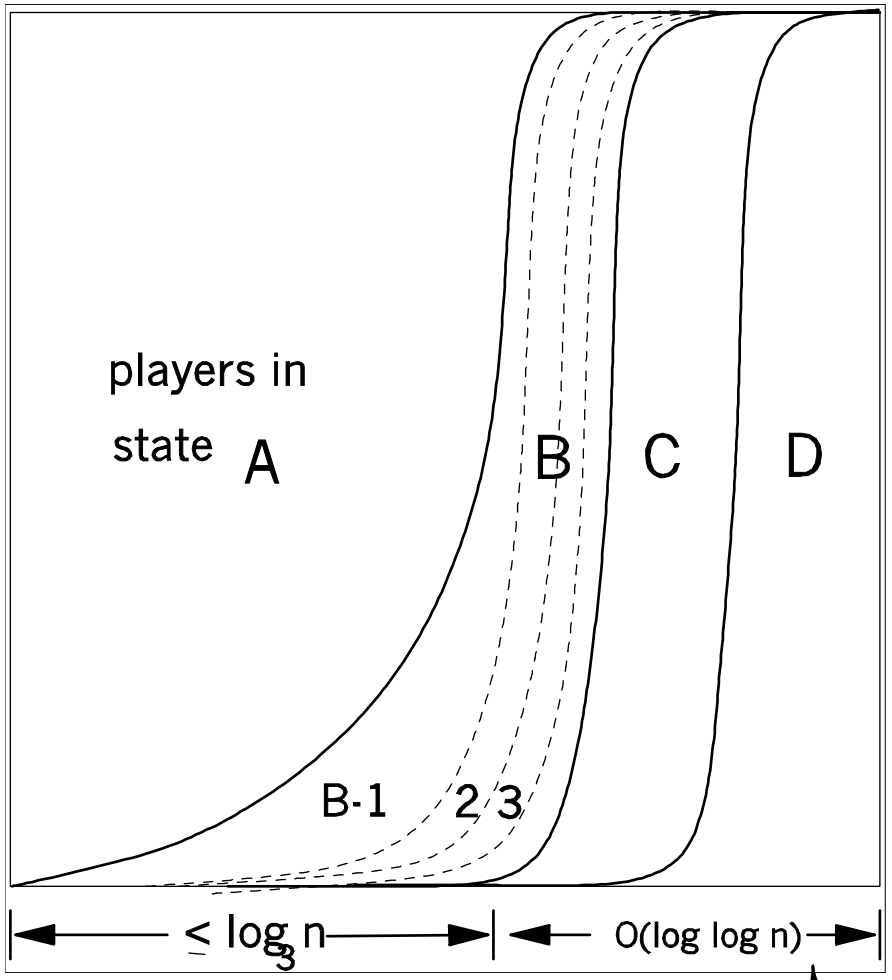
- **Simple termination strategy**
  - If the rumor is older than max_{ctr}, then stop transmission
- **Advantages**
  - simple
- **Disadvantage**
  - Choice of max_{ctr} is critical
  - If max_{ctr} is too small then not all nodes are informed
  - If max_{ctr} is too large, then the message overhead is $\Omega(n\ max_{ctr})$
- **Optimal choice for push-communication**
  - $max_{ctr} = O(\log n)$
  - Number of messages: $O(n \log n)$
- **Pull communication**
  - $max_{ctr} = O(\log n)$
  - Number of messages: $O(n \log n)$
- **Push&Pull communication**
  - $max_{ctr} = \log_3 n + O(\log \log n)$
  - Number of messages: $O(n \log \log n)$

Shenkers Min-Counter Algorithm

- **Only is the rumor is seen as old then contact partners increase the age-counter**
- **Shenkers Min-Counter-Algorithmus für max_{ctr} = O(log log n)**
  - Every player P stores age-variable $ctr_R(P)$ for each rumor R
  - A: player P does not know the rumor:
    - $ctr_R(P) \leftarrow$ **1**
  - B: If player P sees rumor for the first time
    - $ctr_R(P) \leftarrow 1$
  - B: If partners $Q_1, Q_2, \ldots, Q_m$ communicate with P in a round
  - If $min_i\{ctr_R(Q_i)\} \geq ctr_R(P)$ then
    - $ctr_R(P) \leftarrow ctr_R(P) + 1$
  - C: If $ctr_R(P) \geq max_{ctr}$ then
    - tell the rumor for max_{ctr} more rounds
    - then D: stop sending the rumor
- **Theorem**
  - Shenkers Min-Counter algorithms informs all nodes using Push&Pull-communication in $\log_3 n + O(\log \log n)$ rounds with probability $1-n^{-c}$, using at most O(n log log n) messages.

Shenker‘s Min-Counter-Algorithm

- **Theorem**
  - Shenkers Min-Counter algorithms informs all nodes using Push&Pull-communication in $\log_3 n + O(\log \log n)$ rounds with probability $1-n^{-c}$, using at most $O(n \log \log n)$ messages.



Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Game Theory

- **Block based Protocols**
  - Unfairness of Multicast
  - Splitstream
  - Bittorrent
  - Incentives in Bittorrent
- **Game Theory**
  - Nash Equilibria
  - Mechanism Design

Selfish Behavior in P2P

- **Reasons**
  - Psychology of users
  - Lack of central authority
  - Highly dynamic memberships
  - Availability of cheap identities
  - Hidden or untraceable actions
  - Deceitful behavior
- **Implications**
  - Success of P2P networks must take into account economic behavior of users

Typical Features of Peer to Peer Systems

- **Social dilemma**
  - defective behavior (not uploading) is rational behavior, i.e. maximise the utility
- **Asymmetric transactions**
  - a peer wants a service
  - another provides this service
- **Untraceable defections**
  - it is not clear which peer declines a service
- **Dynamic population**
  - peers change the behavior
  - peers enter and leave the system

Incentives for Cooperation

- Inherent generosity
- Monetary payment schemes
- Reciprocity-based schemes

Inherent Generosity

- **Standard model of behavioral economics**
  - based on purely self-interest
  - does not explain all behavior of people
- **User generosity has a great impact on existing peer-to-peer systems**
  - can be determined analytically

Monetary Payment Schemes

- **Golle, Leyton-Brown, Mironov, Lillibridge 2001, „Incentives for Sharing in peer-to-peer Networks“**
  - consider free-rider problem in Napster
  - assume selfish behavior
  - if all peers are selfish this leads to the strict Nash equilibrium
  - introduce micro-payment system to overcome this problem
  - encourage positive behavior by virtual money

Basics of Game Theory

- **Prisoner's dilemma (Flood&Drescher 1950)**
  - two suspects arrested
  - if one testifies and the other remains silent then the witness is released the other serves 10 years prison
  - if both testify then both serve 5 years prison
  - if no one testify then they receive 1/2 year prison
- **Best social strategy**
  - no one testifies
- **Nash equilibrium**
  - for a constant choice of the other party each player optimizes his benefit
  - if both talk then there is a Nash equilibrium

	A talks	A is silent
B talks	A: -5 B: -5	A: -10 B: 0
B is silent	A: 0 B: -10	A: -1/2 B: -1/2

Dominant Strategy

- **Dominant strategy**
  - a strategy is dominant if it is always better than every other strategy
  - in the prisoner‘s dilemma every player has a dominant strategy
    - talk!
- **Nash equilibrium**
  - for a constant choice of the other party each player optimizes his benefit
  - if both talk then there is a Nash equilibrium
  - is not necessary Pareto-optimal

	A talks	A is silent
B talks	A: -5 B: -5	A: -10 B: 0
B is silent	A: 0 B: -10	A: -1/2 B: -1/2

Prisoner‘s Dilemma of Peer to Peer Filesharing

- **Rational strategy for downloading peer:**
  - Download
- **Rational strategy for uploading peer:**
  - Don‘t upload
- **Nash equilibrium**
  - Uploader rejects upload for downloader

	U: Peer uploads	**U: Peer rejects upload**
D: Peer downloads	**D: 10** **U: -1**	**D: 0** **U: 0**
D: Peer does not download	**D: 0** **U: 0**	**D: 0** **U: 0**

Monetary Payment Schemes

- **Advantage**
  - allow to use economic mechanisms
  - charge free-riders for misbehavior
- **Disadvantage**
  - require infrastructure for accounting and micropayments
- **Major problems**
  - how to encourage truthful relevation of costs
    - solution: Vickrey-Clarke-Groves (VCG-mechanisms)
    - strategyproof mechanism
      - encourage truthful revelation in dominant strategies
  - how to encourage cooperate behavior despite hidden actions
    - information asymmetry
    - use contracts
  - how to deliver the payment
    - e.g. the deliverer also receives some part of the payment

Mechanism Design

- **Define rules of the games**
  - such that rational behavior is good behavior
    - e.g. auction system: second best wins
- **Inverse game theory**
  - how to design the rules such that the desired outcome occurs
  - provide incentives
- **Obedient center**
  - the rule system must be enforced on all the nodes
  - altruistic rule maker
  - central control or distributed software control mechanism or cryptography
- **Mechanism design can be computationally hard**
  - calculating the optimal strategy can be difficult
  - not all the information may be available to each player
  - finding the best rule system poses an even more difficult problem
- **Algorithmic Mechanism Design**
  - Mechanism is carried out via a distributed computation

Reciprocity based Schemes

- **Reciprocity based schemes**
  - Users maintain histories of past behavior of other users
  - used for decision making
- **Direct-reprocity scheme**
  - A decides how to serve user B based solely on the service that B has provided
  - e.g. Bittorrent
  - still possibilities for manipulation
- **Indirect-reciprocity scheme**
  - aka. reputation based schemes
  - more scalable for
    - large population sizes
    - highly dynamic memberships
    - infrequent repeat transactions
- **Problems**
  - How to treat newcomers?
    - whitewashing attacks
    - irreplacable pseudonyms
    - penalty for newcomers
  - Indirect reciprocity is vulnerable to deceits, false accusations & false praises
    - sybil attacks
    - sybilproofness

Reciprocative Decision Functions

- **Discriminating Server Selection**
  - use history records to choose partners
- **Shared history**
  - communicate the history with other peers
    - problem: false praise or false accusations
- **Subjective reputation**
  - e.g. max-flow algorithm that collects the reputation be the combination of history of other users
  - e.g. page-rank algorithm
- **Adaptive stranger policy**
  - treat strangers like the previously seen strangers
    - arrest usual suspects only if the crime rate is high
- **Short-term history**
  - long history records allow peers to gather reputation and then turn into traitors
  - short-term history records will discipline all peers

Future Research Directions

- **How to overcome the prisoner‘s dilemma**
  - game theory the right tool?
- **What is rational behavior?**
  - Is Nash equilibrium the right model
- **Influence of different user behavior**
  - different grades of selfishness or altruism
- **Contracts can lead to desired behavior of peers**
  - computational complexity of optimal contracts unknown

Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Network Coding

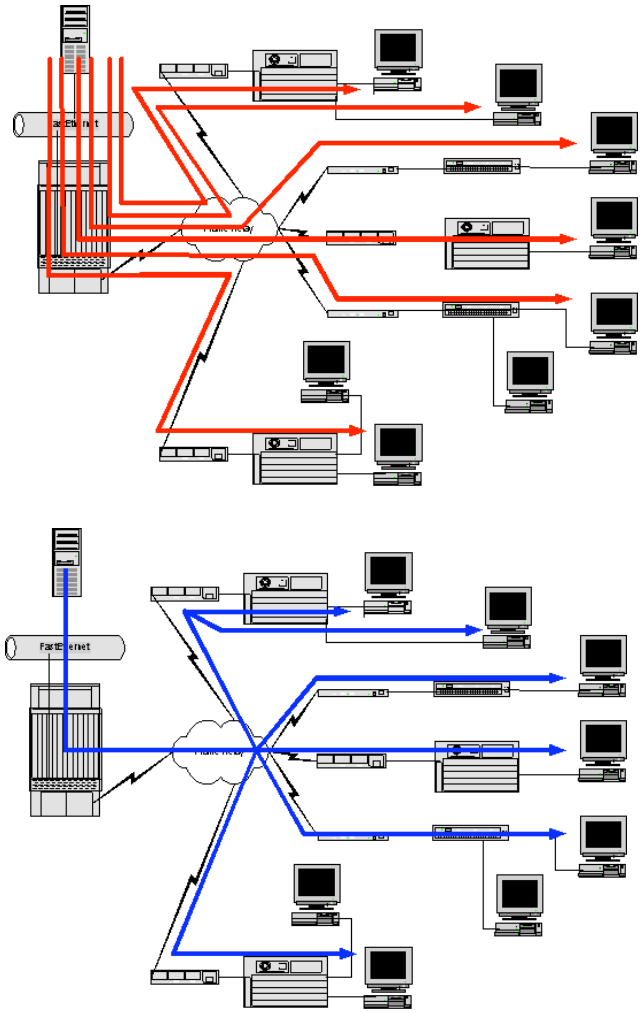
- **Principle**
  - Method
  - Application
- **Practical Network Coding**
  - How to code
  - Avalanche
- **Sparse Codes**
  - FEC
  - Paircoding

Peer to Peer Networks

Fast Download

IP Multicast

- **Motivation**
  - Transmission of a data stream to many receivers
- **Unicast**
  - For each stream message have to be sent separately
  - Bottleneck at sender
- **Multicast**
  - Stream multiplies messages
  - No bottleneck



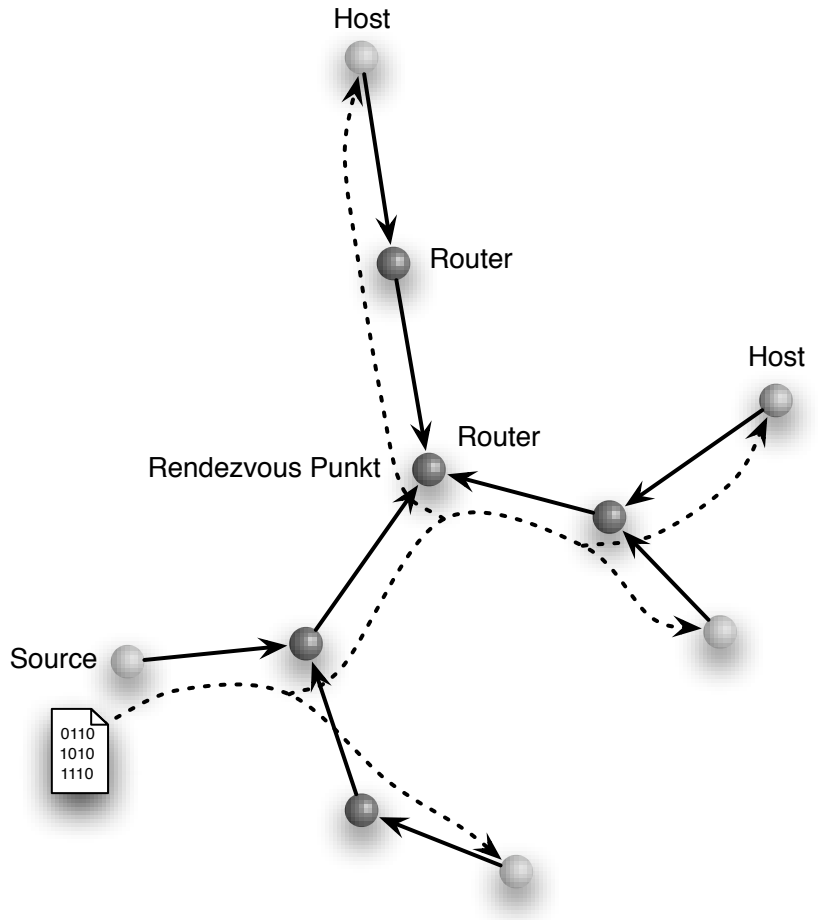
Peter J. Welcher
www.netcraftsmen.net/.../ papers/multicast01.html

Working Principle

- **IPv4 Multicast Addresses**
  - class D
    - outside of CIDR (Classless Interdomain Routing)
  - 224.0.0.0 - 239.255.255.255
- **Hosts register via IGMP at this address**
  - IGMP = Internet Group Management Protocol
  - After registration the multicast tree is updated
- **Source sends to multicast address**
  - Routers duplicate messages
  - and distribute them into sub-trees
- **All registered hosts receive these messages**
  - ends after Time-Out
  - or when they unsubscribe
- **Problems**
  - No TCP only UDP
  - Many routers do not deliver multicast messages
    - solution: tunnels

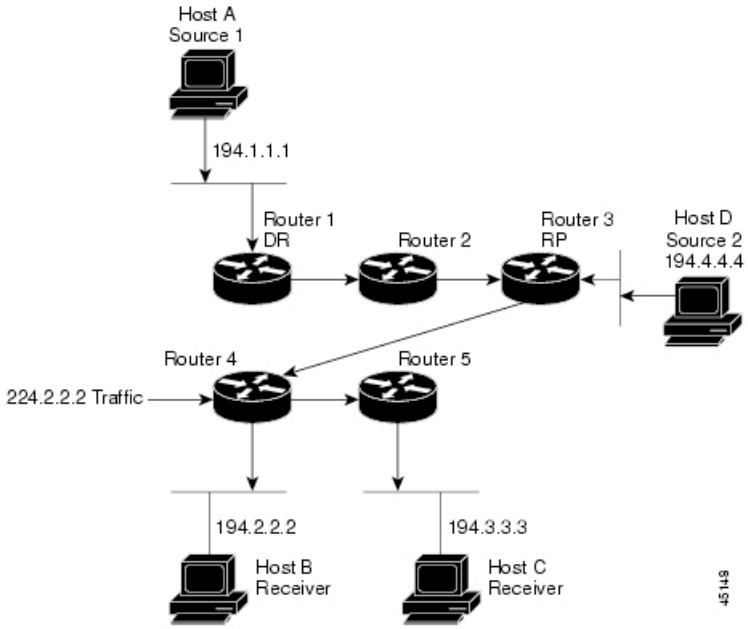
Routing Protocols

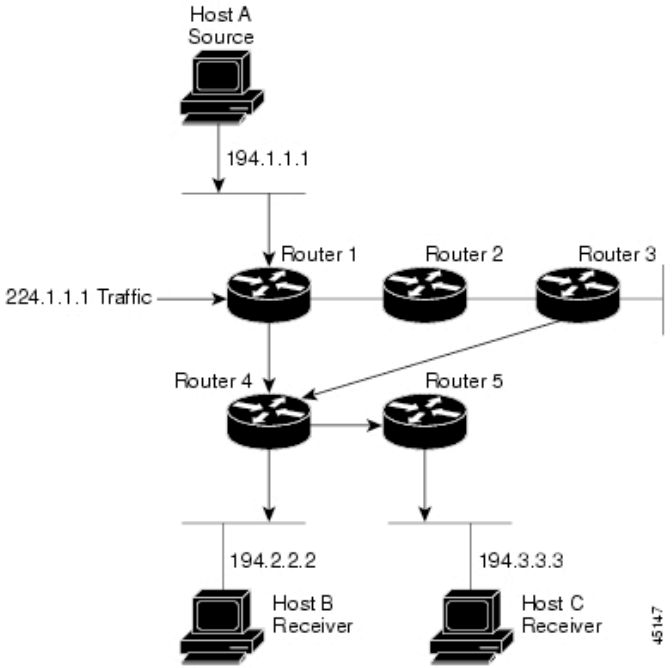
- **Distance Vector Multicast Routing Protocol (DVMRP)**
  - used for years in MBONE
    - particularly in Freiburg
  - own routing tables for multicast
- **Protocol Independent Multicast (PIM)**
  - in Sparse Mode (PIM-SM)
  - current (de facto) standard
  - prunes multicast tree
  - uses Unicast routing tables
  - is more independent from the routers
- **Prerequisites of PIM-SM:**
  - needs Rendezvous-Point (RP) in one hop distance
  - RP must provide PIM-SM
  - or tunneling to a proxy in the vicinity of the RP



PIM-SM
Tree Construction

- **Host A Shortest-Path-Tree**
- **Shared Distribution Tree**

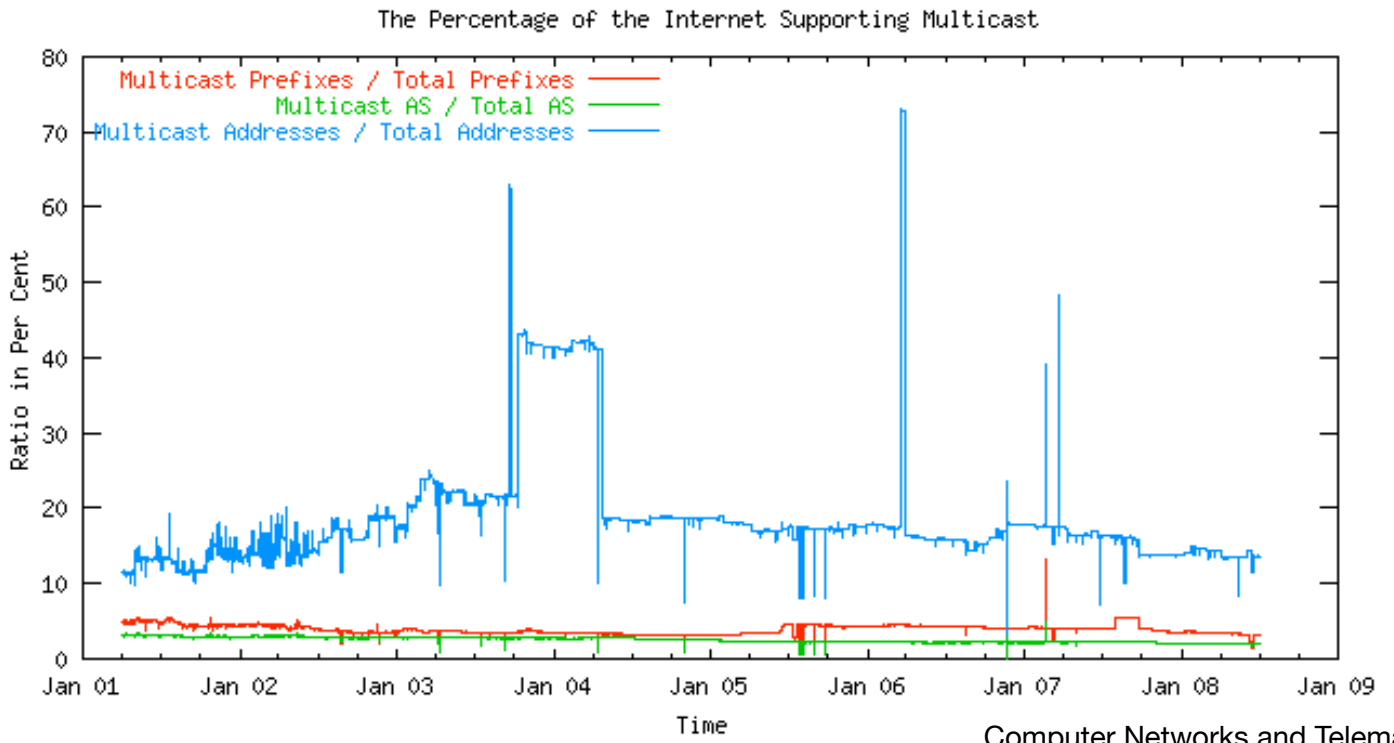




From Cisco: http://www.cisco.com/en/US/products/hw/switches/ps646/products_configuration_guide_chapter09186a008014f350.html

IP Multicast Seldomly Available

- **IP Multicast is the fastest download method**
- **Yet, not many routers support IP multicast**
  - http://www.multicasttech.com/status/

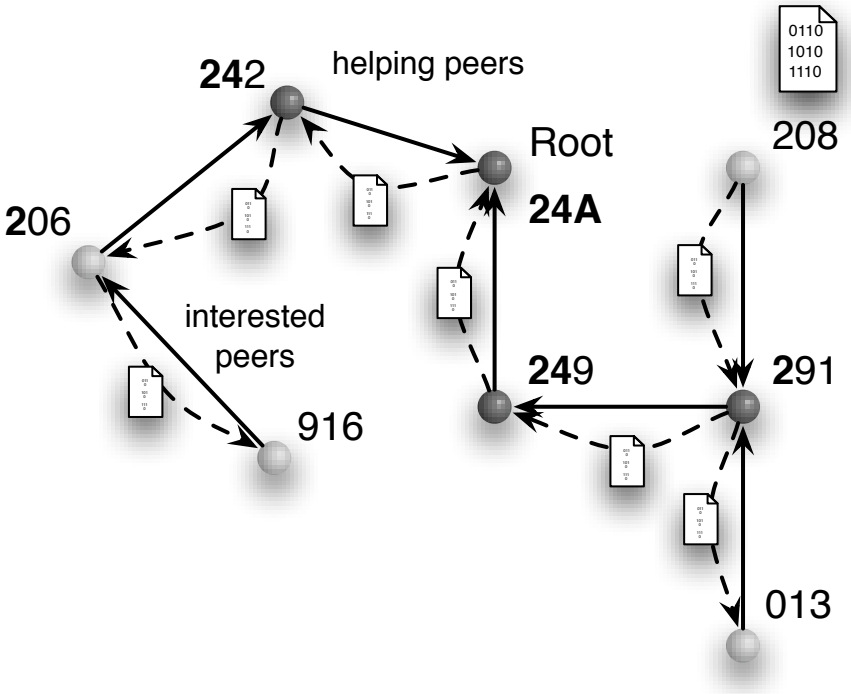


Why so few Multicast Routers?

- **Despite successful use**
  - in video transmission of IETF-meetings
  - MBONE (Multicast Backbone)
- **Only few ISPs provide IP Multicast**
- **Additional maintenance**
  - difficult to configure
  - competing protocols
- **Enabling of Denial-of-Service-Attacks**
  - Implications larger than for Unicast
- **Transport protocol**
  - only UDP
    - Unreliable
  - Forward error correction necessary
    - or proprietary protocols at the routers (z.B. CISCO)
- **Market situation**
  - consumers seldomly ask for multicast
    - prefer P2P networks
  - because of a few number of files and small number of interested parties the multicast is not desirable (for the ISP)
    - small number of addresses

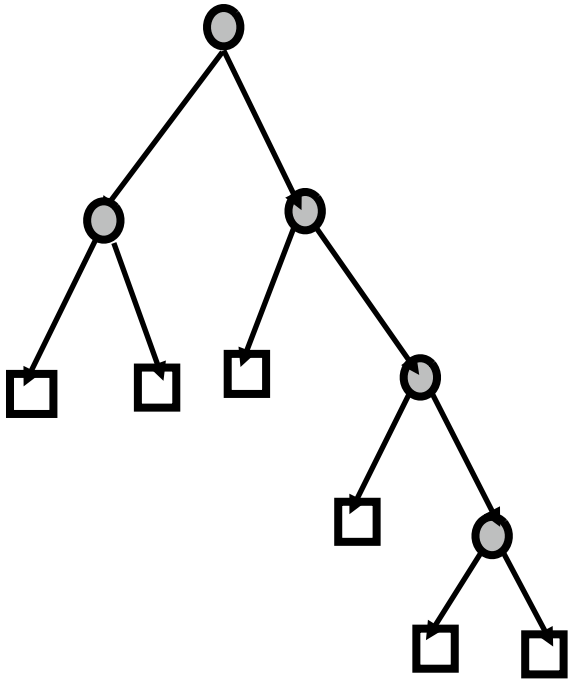
Scribe & Friends

- **Multicast-Tree in the Overlay Network**
- **Scribe [2001] is based on Pastry**
  - Castro, Druschel, Kermarrec, Rowstron
- **Similar approaches**
  - CAN Multicast [2001] based on CAN
  - Bayeux [2001] based on Tapestry
- **Further approaches**
  - Overcast [´00] and Narada [´00]
  - construct multi-cast trees using unicast connections
  - do not scale



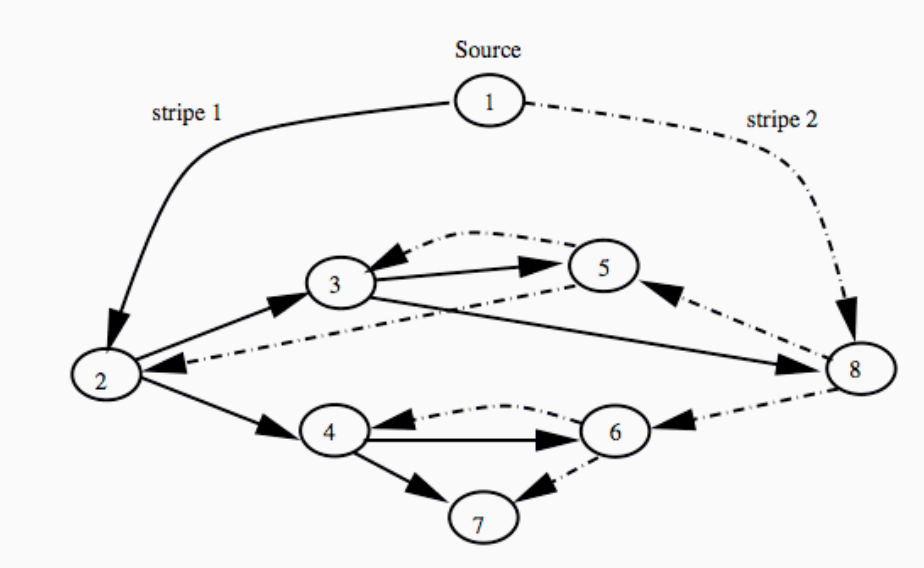
Split-Stream Motivation

- **Multicast trees discriminate certain nodes**
- **Lemma**
  - In every binary tree the number of leaves = number of internal nodes +1
- **Conclusion**
  - Nearly half of the nodes distribute data
  - While the other half does not distribute any data
  - An internal node has twice the upload as the average peer
- **Solution: Larger degree?**
- **Lemma**
  - In every node with degree d the number of internal nodes k und leaves b we observe
    - (d-1) k = b -1
- **Implication**
  - Less peers have to suffer more upload



Split-Stream

- **Castro, Druschel, Kermarrec, Nandi, Rowstron, Singh 2001**
- **Idea**
  - Partition a file of size into k small parts
  - For each part use another multicast tree
  - Every peer works as leave and as distributing internal tree node
    - except the source
- **Ideally, the upload of each node is at most the download**



Bittorrent

- **Bram Cohen**
- **Bittorrent is a real (very successful) peer-to-peer network**
  - concentrates on download
  - uses (implicitly) multicast trees for the distribution of the parts of a file
- **Protocol is peer oriented and not data oriented**
- **Goals**
  - efficient download of a file using the uploads of all participating peers
  - efficient usage of upload
    - usually upload is the bottleneck
    - e.g. asymmetric protocols like ISDN or DSL
  - fairness among peers
    - seeders against leeches
  - usage of several sources

Bittorrent
Coordination and File

- **Central coordination**
  - by tracker host
  - for each file the tracker outputs a set of random peers from the set of participating peers
    - in addition hash-code of the file contents and other control information
  - tracker hosts to not store files
    - yet, providing a tracker file on a tracker host can have legal consequences
- **File**
  - is partitions in smaller pieces
    - as describec in tracker file
  - every participating peer can redistribute downloaded parts as soon as he received it
  - Bittorrent aims at the Split-Stream idea
- **Interaction between the peers**
  - two peers exchange their information about existing parts
  - according to the policy of Bittorrent outstanding parts are transmitted to the other peer

Bittorrent
Part Selection

- **Problem**
  - The Coupon-Collector-Problem is the reason for a uneven distribution of parts
    - if a completely random choice is used
- **Measures**
  - Rarest First
    - Every peer tries to download the parts which are rarest
      - density is deduced from the comunication with other peers (or tracker host)
    - in case the source is not available this increases the chances the peers can complete the download
  - Random First (exception for new peers)
    - When peer starts it asks for a random part
    - Then the demand for seldom peers is reduced
      - especially when peers only shortly join
  - Endgame Mode
    - if nearly all parts have been loaded the downloading peers asks more connected peers for the missing parts
    - then a slow peer can not stall the last download

Bittorrent Policy

- **Goal**
  - self organizing system
  - good (uploading, seeding) peers are rewarded
  - bad (downloading, leeching) peers are penalized
- **Reward**
  - good download speed
  - un-choking
- **Penalty**
  - Choking of the bandwidth
- **Evaluation**
  - Every peers Peers evaluates his environment from his past experiences

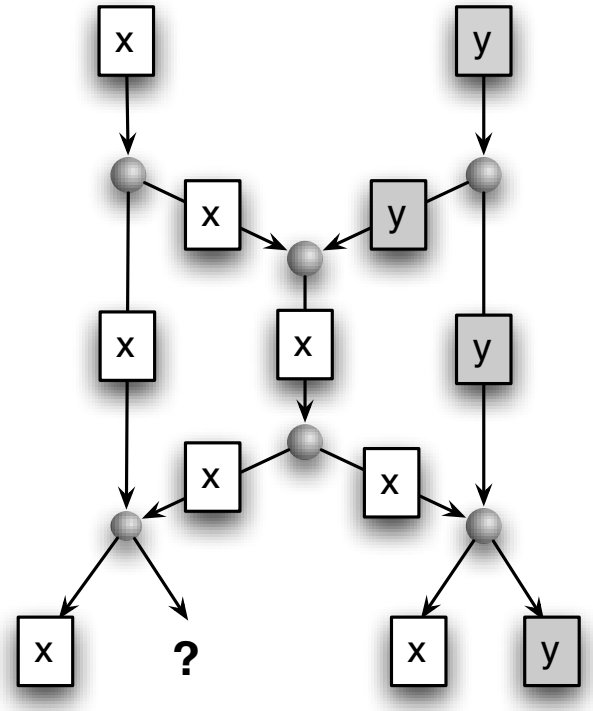
Bittorrent Choking

- **Every peer has a choke list**
  - requests of choked peers are not served for some time
  - peers can be unchoked after some time
- **Adding to the choke list**
  - Each peer has a fixed minimum amount of choked peers (e.g. 4)
  - Peers with the worst upload are added to the choke list
    - and replace better peers
- **Optimistic Unchoking**
  - Arbitrarily a candidate is removed from the list of choking candidates
    - the prevents maltreating a peer with a bad bandwidth

Network Coding

- **R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", (IEEE Transactions on Information Theory, IT-46, pp. 1204-1216, 2000)**

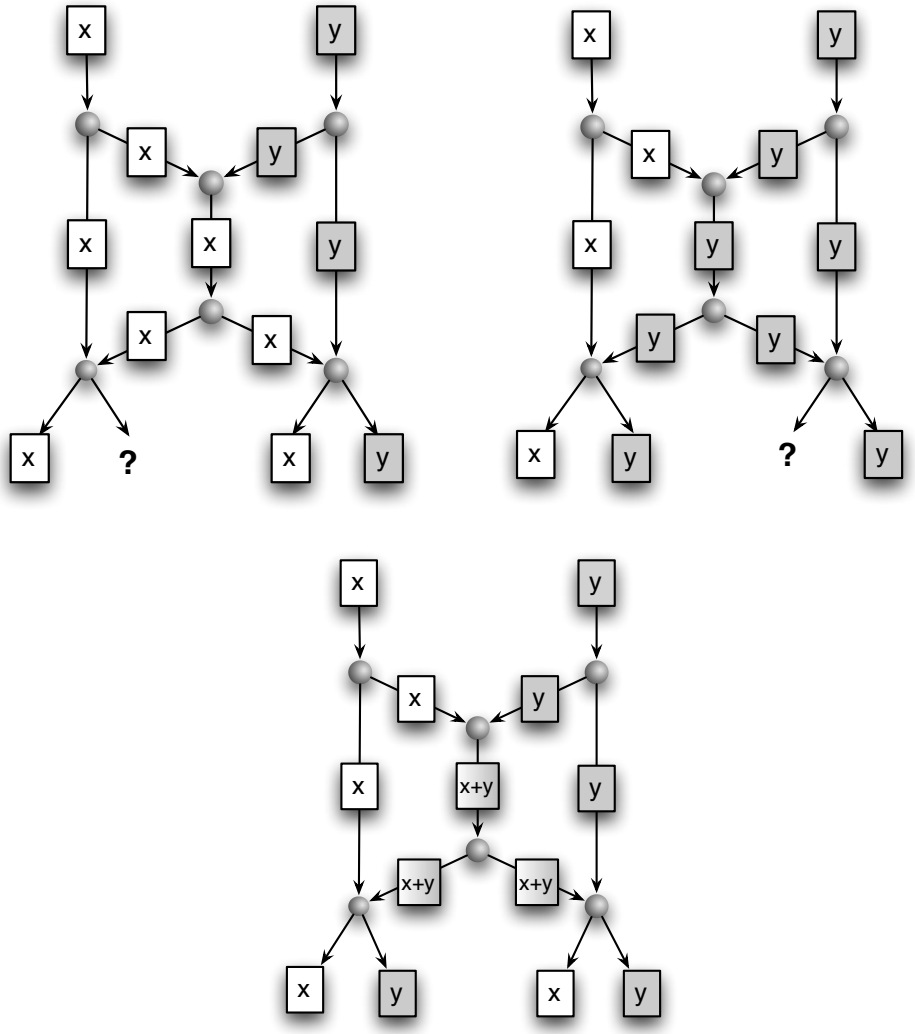
- **Example**
  - Bits x and y need to be transmitted
  - Every line transmits one bit
  - If only bits are transmitted
    - then only x or y can be transmitted in the middle?
  - By using X we can have both results at the outputs



Network Coding

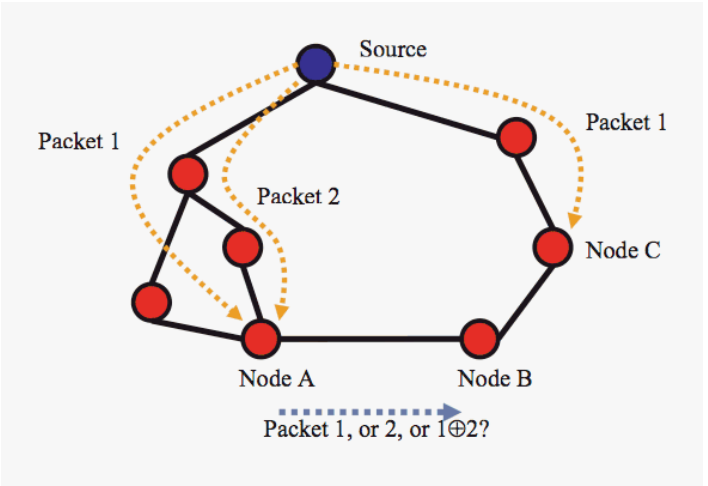
- **R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", (IEEE Transactions on Information Theory, IT-46, pp. 1204-1216, 2000)**

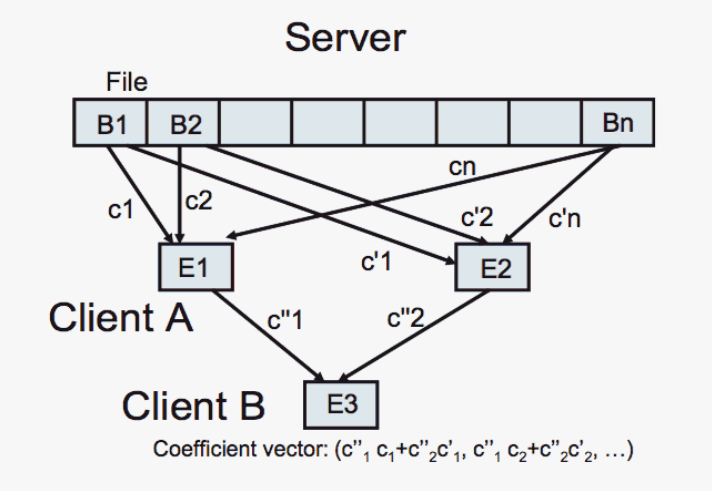
- **Theorem [Ahlswede et al.]**
  - There is a network code for each graph such that each node receives as much information as the maximum flow of the corresponding flow problem



Practical Network Coding Avalanche

- **Christos Gkantsidis, Pablo Rodriguez Rodriguez, 2005**
- **Goal**
  - Overcoming the Coupon-Collector-Problem
    - a file of m parts can be always reconstructed if at least m network codes have been received
  - Optimal transmission of files within the available bandwidth
- **Method**
  - Use codes as linear combinations of a file
    - Produced code contains the vector and the variables
  - During the distribution the linear combination are re-combined to new parts
  - The receiver collects the linear combinations
  - and reconstructs the original file using matrix operations





Coding and Decoding

- File: $x_1, x_2, \ldots, x_m$
- Codes: $y_1, y_2, \ldots, y_m$
- Random Variables r_{ij}

$$(r_{i1} r_{i2} \ldots r_{im}) \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = y_i$$

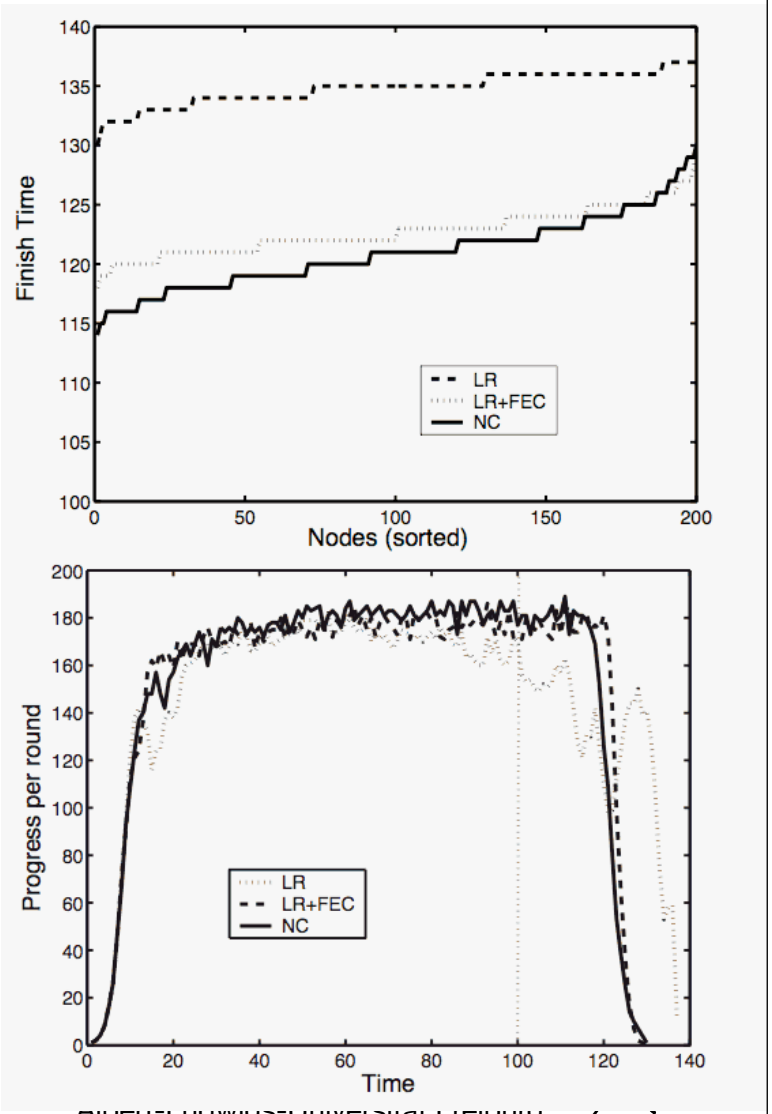
$$\begin{pmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mm} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

- If the matrix is invertable then

$$\begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mm} \end{pmatrix}^{-1} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

Speed of Network-Coding

- **Comparison**
  - Network-Coding (NC) versus
  - Local-Rarest (LR) and
  - Local-Rarest+Forward-Error-Correction (LR+FEC)



Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Anonymity & Security

- **Anonymity**
  - Freenet
  - Onion Routing
  - Secret Sharing
  - Dining Cryptographers
- **Security**
  - Byzantine Generals
  - Cuckoo Hashing
- **Dark rooms**
  - Self-Encrypted software
  - Shibboleth
  - Crypto-Layers

Attacks

- **Denial-of-Service Attacks (DoS)**
  - or distributed denial of service attacks (DDoS)
  - one or many peers ask for a document
  - peers are slowed down or blocked completely
- **Sybil Attacks**
  - one attacker produces many fake peers under new IP addresses
  - or the attacker controls a bot-net
- **Use of protocol weaknesses**
- **Infiltration by malign peers**
  - Byzantine Generals
- **Timing attacks**
  - messages are slowed down
  - communication line is slowed down
  - a connection between sender and receiver can be established
- **Poisoning Attacks**
  - provide false information
  - wrong routing tables, wrong index files etc.
- **Eclipse Attack**
  - attack the environment of a peer
  - disconnect the peer
  - build a fake environment

Motivation

- **Society**
  - Free speech is only possible if the speaker does not suffer negative consequences
  - Thus, only an anonymous speaker has truly free speech
- **Copyright infringement**
  - Copying items is the best (and most) a computer can do
  - Copyright laws restrict copying
  - Users of file sharing systems do not want to be penalized for their participation or behavior
- **Dictatorships**
  - A prerequisite for any oppressing system is the control of information and opinions
  - Authors, journalists, civil rights activists like all citizens should be able to openly publish documents without the fear of penalty
- **Democracies**
  - In many democratic states certain statements or documents are illegitimate, e.g.
    - (anti-) religious statements
    - insults (against the royalty)
    - certain sexual contents
    - political statements (e.g. for fascism, communism, separation, revolution)
- **A anonymizing P2P network should secure the privacy and anonymity of each user without endangering other users**

Cryptography in a Nutshelf

- **Symmetric Cryptography**
  - AES
  - Affine Cryptosystems
- **Public-Key Cryptography**
  - RSA
  - ElGamal
- **Digital Signatures**
- **Public-Key-Exchange**
  - Diffie-Hellman
- **Interactive Proof Systems**
  - Zero-Knowledge-Proofs
  - Secret Sharing
  - Secure Multi-Party Computation

Diffie-Hellman Key Exchange

- **Diffie, Hellman 1978**
- **Goal**
  - Exchange a secret key between two participants A and B while all communication is surveilled
- **Initiator A**
  - choose prime number p
  - and a generator g
    - i.e. 1, g, g^2, ..., g^{p-2} mod p are all different
  - publishes p and g
- **Key Exchange**
  - A picks a random number a
  - B picks a random number b
  - A sends $x=g^a \bmod p$
  - B sends $y=g^b \bmod p$
  - A computes $K=y^a \bmod p$
  - B computes $K=x^b \bmod p$
- **Both parties know $K = x^{ab} \bmod p$**
- **Scheme relies on the difficulty to compute the discrete logarithm mod p**
  - Now A and B can use K as secret key for symmetric cryptography

Zero-Knowledge Proofs

- **Without revealing secret information it is possible to prove the knowledge of a fact to a partner**
- **For this an interactive protocol is used**
- **Two parties**
  - The prover
  - The verifier
- **Correctness of the protocol follows when the verifier can reproduce the results without the help of the prover**
- **There are Zero-Knowledge proofs for all problems in PSPACE**
  - „IP = PSPACE“, Adi Shamir 1992

Zero-Knowledge Proofs for Hamiltonian Cycles

- **Both parties know a graph G**
- **The prover P knows a hamiltonian cycle**
- **The verifier V asks P for a proof**
  - but P does not want show his solution
- **Perform the following rounds for some time until V is convinced**
  - P renumbers all nodes and produces a new graph H with the same edges and the corresponding hamiltonian cycle
  - P sends the new graph to V
  - V asks one of the following questions to P
    - Prove that G is isomorphic to H
      - i.e. the same but renumbered graph
      - then V sends the renumbering table
    - Proof that you have a Hamiltonian cycle in H
      - then V sends the new Hamiltonian cycle
  - V checks the correctness of each case but does not know the Hamiltonian cycle in the original graph

Peer-to-Peer Networks

Secret Sharing 1979

Blakley's Secret Sharing

- **George Blakley, 1979**
- **Task**
  - n persons have to share a secret
  - only when k of n persons are present the secret is allowed to be revealed
- **Blakley's scheme**
  - in a k-dimensional space the intersection of k non-parallel k-1-dimensional spaces define a point
  - this point is the information
  - with k-1 sub-spaces one gets only a line
- **Construction**
  - A third (trusted) instance generate for a point n in R^k k non-parallel k-1-dimensional hyper-spaces

Shamir's Secret Sharing Systems

- **Adi Shamir, 1979**
- **Task**
  - n persons have to share a secret s
  - only k out of n persons should be able to reveal this secret
- **Construction of a trusted third party**
  - chooses random numbers $a_1,\ldots,a_{k-1}$
  - defines

$$f(x) = s + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$$

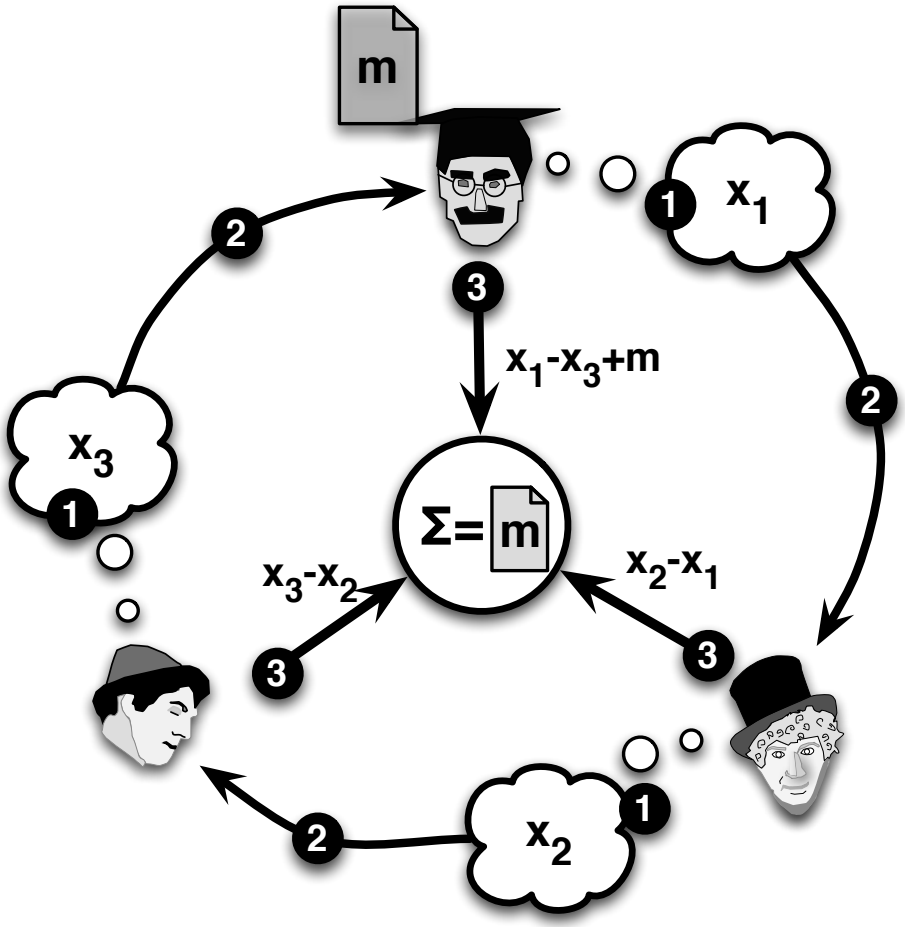
  - chooses random $x_1, x_2, \ldots, x_n$
  - sends $(x_i, f(x_i))$ to player i
- **If k persons meet**
  - then they can compute the function f by the fundamental theorem of algebra
    - a polynomial of degree d is determined by d+1 values
  - for this they exchange their values and compute by interpolation
    - (e.g. using Lagrange polynoms)
- **If k-1 persons meet**
  - they cannot compute the secret at all
  - every value of s remains possible
- **Usually, Shamir's and Blakley's scheme are used in finite fields**
  - i.e. Galois fields (known from CRC)
  - this simplifies the computation and avoids rounding errors in the context of floating numbers

Peer-to-Peer Networks

Chaum‘s Dining Cryptographers 1988

Dining Cryptographers

- **Anonymous publications without any tracing possibility**
- **n ≥ 3 cryptographers sit at a round table**
  - neighbored cryptographers can communicate secretly
- **Each peer chooses secret number x_i and communicates it to the right neighbor**
- **If i wants to send a message m**
  - he publishes $s_i = x_i - x_{i-1} + m$
- **else**
  - he publishes $s_i = x_i - x_{i-1}$
- **Now they compute the sum $s=s_1+...+s_n$**
  - if s=0 then there is no message
  - else the sum of all messages

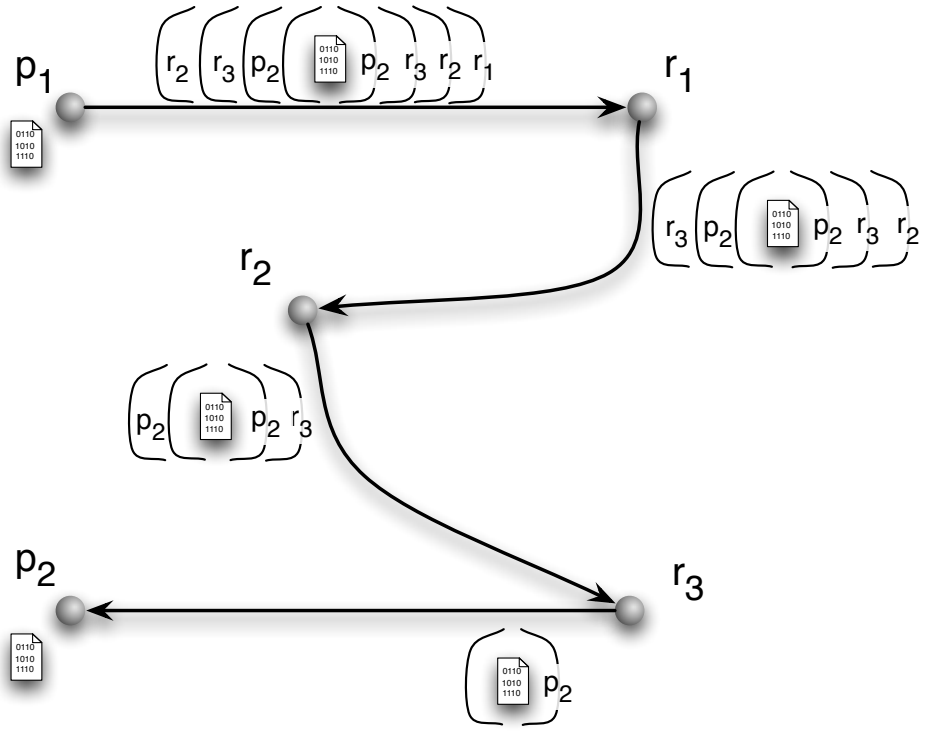


Peer-to-Peer Networks

Chaum Mixes 1981

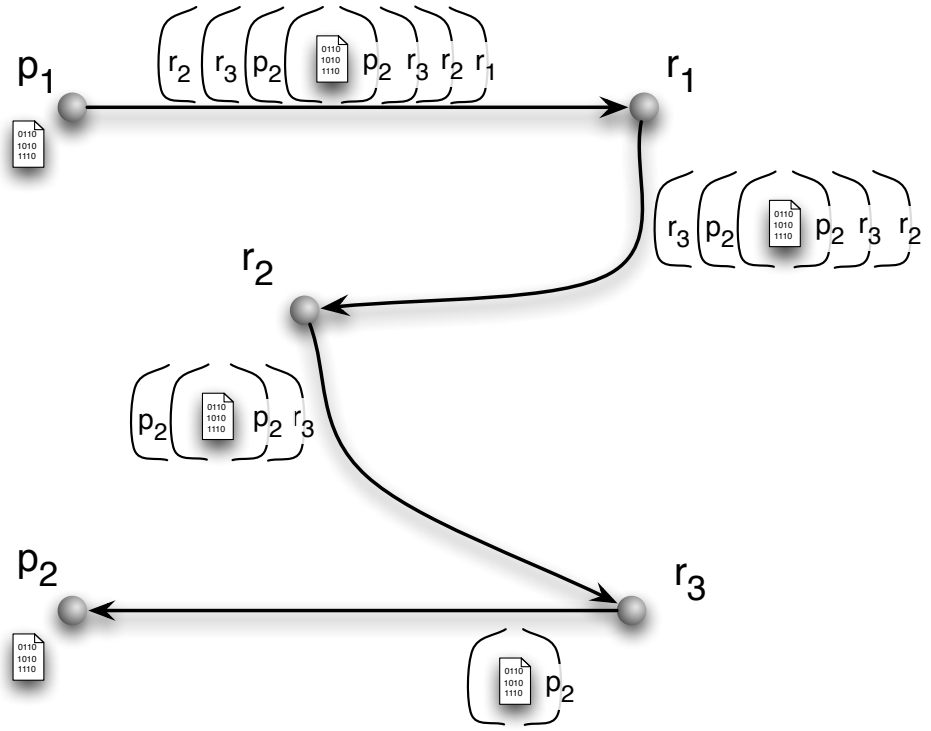
Chaum's Mix-Cascades

- **All peers**
  - publish the public keys
  - are known in the network
- **The sender p_1 now chooses a route**
  - p_1, r_1, r_2, r_3, ..., p_2
- **The sender encrypts m according to the public keys from**
  - p_2, ... r_3, r_2, r_1
  - and sends the message
  - $f(pk_{k1},(r_2,f(pk_{r2}...f(pk_{rk},(p_2,f(pk_{p2},m)))...)))$
  - to r_1
- **r_1 encrypts the code, deciphers the next hop r_2 and sends it to him**
- **...**
- **until p_2 receives the message and deciphers it**



Chaum‘s Mix Cascades

- **No peer on the route**
  - knows its position on the route
  - can decrypt the message
  - knows the final destination
- **The receiver does not know the sender**
- **In addition peers may voluntarily add detour routes to the message**
- **Chaum‘s Mix Cascades**
  - aka. Mix Networks or Mixes
  - is safe against all sort of attacks,
  - but not against traffic analysis



Peer-to-Peer Networks

Onion Routing
1998

TOR - Onion Routers

- **David Goldschlag, Michael Reed, and Paul Syverson, 1998**
- **Goal**
  - Preserve private sphere of sender and receiver of a message
  - Safety of the transmitted message
- **Prerequisite**
  - special infrastructure (Onion Routers)
    - all except some smaller number of exceptions cooperate
- **Method**
  - Mix Cascades (Chaum)
  - Message is sent from source to the target using proxies (Onion Routers)
  - Onion Routers unpredictably choose other routers as intermediate routers
  - Between sender, Onion Routers, and receiver the message is encrypted using symmetric cryptography
  - Every Onion Router only knows the next station
  - The message is encoded like an onion
- **TOR is meant as an infrastructure improvement of the Internet**
  - not meant as a peer-to-peer network
  - yet, often used from peer-to-peer networks

Other Work based on Onion Routing

- **Crowds**
  - Reiter & Rubin 1997
  - anonymous web-surfing based on Onion Routers
- **Hordes**
  - Shields, Levine 2000
  - uses sub-groups to improve Onion Routing
- **Tarzan**
  - Freedman, 2002
  - A Peer-to-Peer Anonymizing Network Layer
  - uses UDP messages and Chaum Mixes in group to anonymize Internet traffic
  - adds fake traffic against timing attacks

Free-Haven

- **Dingledine, Freedman, Molnar, Sniffen Kamin 2000**
- **Goal**
  - Peer-to-Peer based Distributed Data Storage robust against attack from strong adversaries
  - Attacker tries to destroy data
- **Design**
  - Community of servers providing storage
  - Documents are stored using secret sharing on the servers
  - Document shares are exchanged between servers
  - For retrieval the shares are collected (using secured communication) from the storing servers
  - The addresses of the servers storing the data shares must be asked at a separate peer
    - which does not know the content nor the original keyword
  - All communication based on Onion Routing

Free-Haven

- **Operations**
  - Uploading documents
  - Downloading documents
    - preserving the authenticity of the documents
  - Expiration date for documents
    - until then a document is safe
    - then it must be erased
  - Adding new peer servers
  - Mechanisms to deteckt inactive or dead servers

- **Free-Haven**
  - provides anonymity to publishers, readers, servers and to the document
  - no query anonymity
    - others might know the topics a peer is interested
  - relies on trustable servers

Morphmix

- **Peer-to-Morphmix**
  - Rennhard, Plattner 2002
  - P2P-server system for anonymous web-surfing
  - encrypts with symmetric keys for efficiency
  - secret-key exchange with Diffie-Hellman public-key exchange using two additional nodes

- **Peer a uses peer b and w from its neighborhood to prepare connection between a and c**

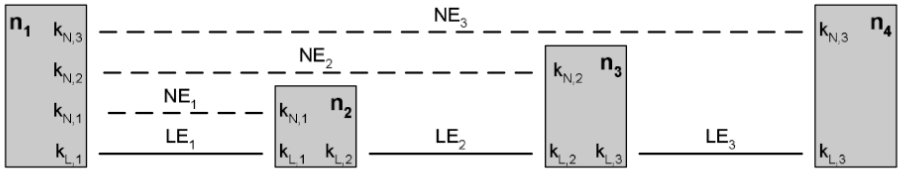


Figure 1: Layers of encryption.

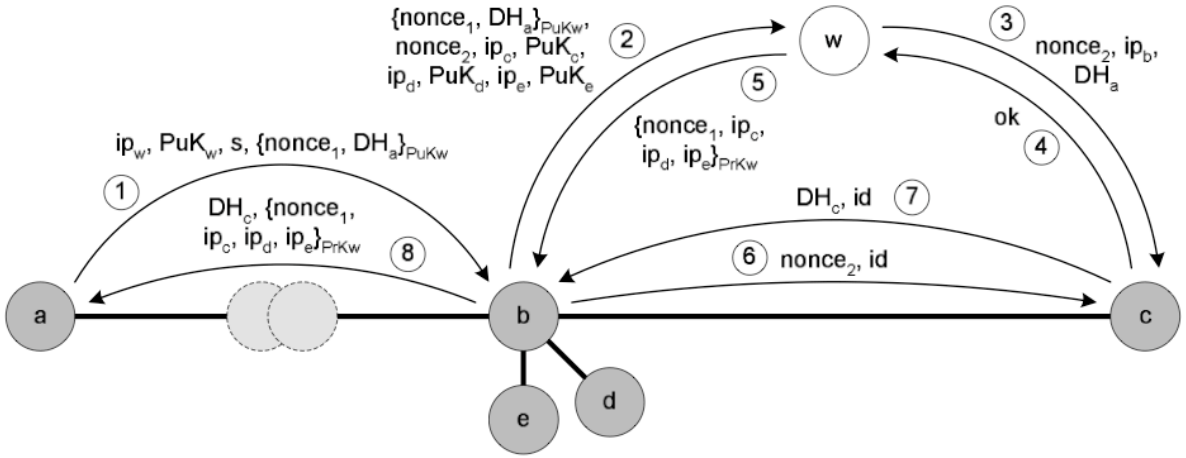


Figure 2: Setting up the nested encryption

Dark-Net & Friend-to-Friend

- **Dark-Net is a private Peer-to-Peer Network**
  - Members can trust all other members
  - E.g.
    - friends (in real life)
    - sports club
- **Dark-Net control access by**
  - secret addresses,
  - secret software,
  - authentication using password, or
  - central authentication

- **Example:**
  - WASTE
    - P2P-Filesharing up to 50 members
    - by Nullsoft (Gnutella)
  - CSpace
    - using Kademlia

Attacks

- **Denial-of-Service Attacks (DoS)**
  - or distributed denial of service attacks (DDoS)
  - one or many peers ask for a document
  - peers are slowed down or blocked completely
- **Sybil Attacks**
  - one attacker produces many fake peers under new IP addresses
  - or the attacker controls a bot-net
- **Use of protocol weaknesses**
- **Infiltration by malign peers**
  - Byzantine Generals
- **Timing attacks**
  - messages are slowed down
  - communication line is slowed down
  - a connection between sender and receiver can be established
- **Poisoning Attacks**
  - provide false information
  - wrong routing tables, wrong index files etc.
- **Eclipse Attack**
  - attack the environment of a peer
  - disconnect the peer
  - build a fake environment

Solutions to the Sybil Attack

- Survey paper by Levine, Shields, Margonin, 2006
- **Trusted certification**
  - only approach to completely eleminate Sybil attacks
    - according to Douceur
  - relies on centralized authority
- **No solution**
  - know the problem and deal with the consequences
- **Resource testing**
  - real world friends
  - test for real hardware or addresses
    - e.g. heterogeneous IP addresses
  - check for storing ability
- **Recurring cost and fees**
  - give the peers a periodic task to find out whether there is real hardware behind each peer
    - wasteful use of resources
  - charge each peer a fee to join the network
- **Trusted devices**
  - use special hardware devices which allow to connect to the network

Solutions to the Sybil Attack

- Survey paper by Levine, Shields, Margonin, 2006
- **In Mobile Networks**
  - use observations of the mobile node
    - e.g. GPS location, neighbor nodes, etc.
- **Auditing**
  - perform tests on suspicious nodes
  - or reward a peer who proves that it is not a clone peer
- **Reputation Systems**
  - assign each peer a reputation which grows over the time with each positive fact
  - the reputation indicates that this peer might behave nice in the future
  - Disadvantage:
    - peers might pretend to behave honestly to increase their reputation and change their behavior in certain situations
    - problem of Byzantine behavior

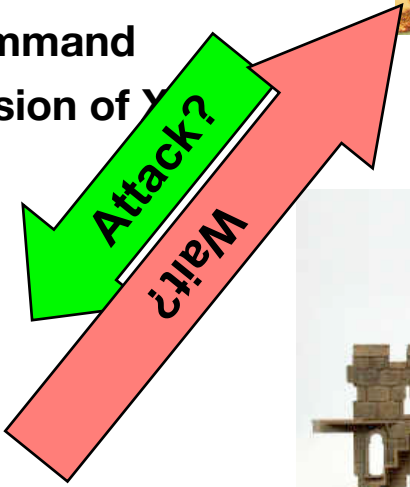
The Problem of Byzantine Generals

- 3 armies prepare to attack a castle
- They are separated and communicate by messengers
- If one army attacks alone, it loses
- If two armies attack, they win
- If nobody attacks the castle is besieged and they win
- One general is a renegade
  - nobody knows who

The Problem of Byzantine Generals

- **The evil general X tries**
  - to convince A to attack
  - to convince B to wait
- **A tells B about X's command**
- **B tells B about his version of X's command**
  - contradiction
- **But is A, B, or X lying?**

A



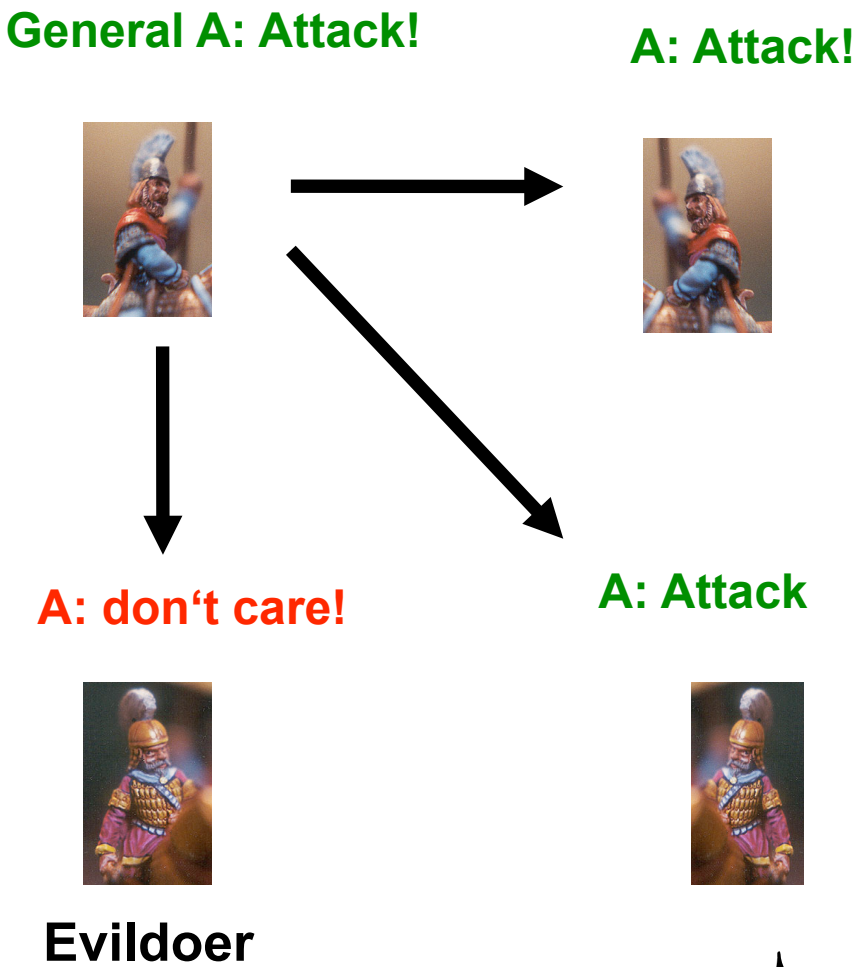
X

B

Wait!

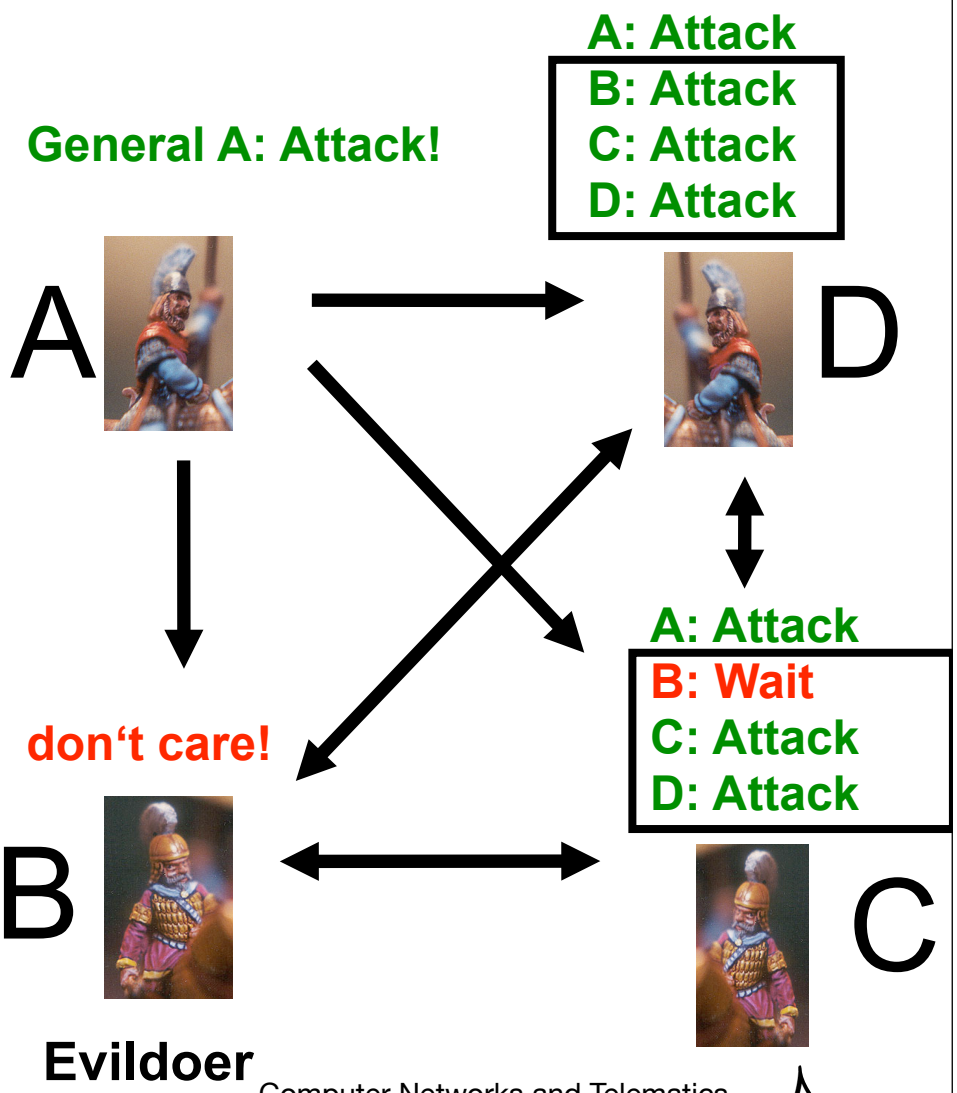
Byzantine Agreement

- **Theorem**
  - The problem of three byzantine generals cannot be solved (without cryptography)
  - It can be solved for 4 generals



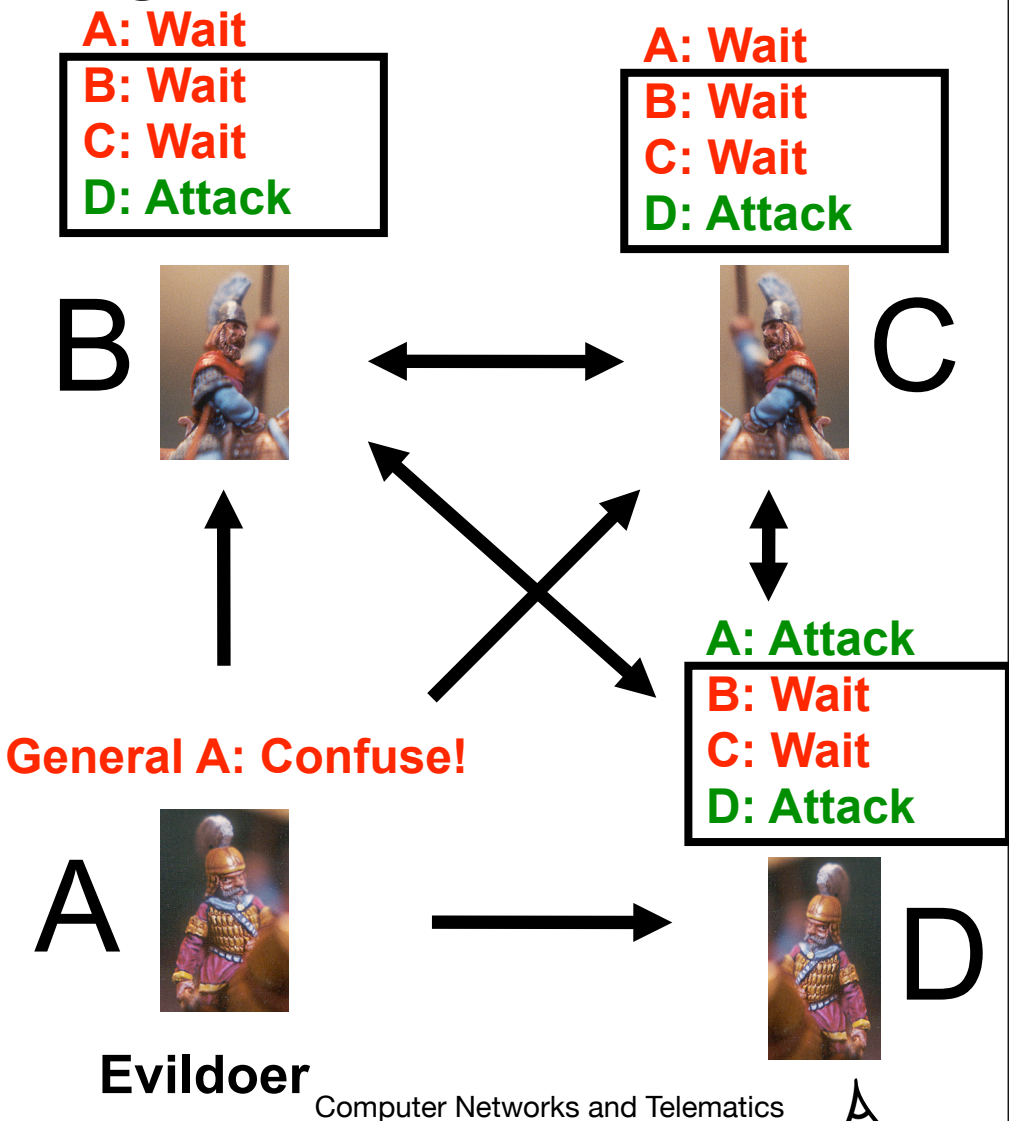
Byzantine Agreement

- **Theorem**
  - The problem of four byzantine generals can be solved (without cryptography)
- **Algorithm**
  - General A sends his command to all other generals
    - A sticks to his command if he is honest
  - All other generals forward the received command to all other generals
  - Every generals computes the majority decision of the received commands and follows this command



Byzantine Agreement

- **Theorem**
  - The problem of four byzantine generals can be solved (without cryptography)
- **Algorithm**
  - General A sends his command to all other generals
    - A sticks to his command if he is honest
  - All other generals forward the received command to all other generals
  - Every generals computes the majority decision of the received commands and follows this command



General Solution of Byzantine Agreement

- **Theorem**
  - If m generals are traitors then 2m+1 generals must be honest to get a Byzantine Agreement
- **This bound is sharp if one does not rely on cryptography**
- **Theorem**
  - If a digital signature scheme is working, then an arbitrarily large number of betraying generals can be dealt with
- **Solution**
  - Every general signs his command
  - All commands are shared together with the signature
  - Inconsistent commands can be detected
  - The evildoer can be exposed

Cuckoo Hashing for Security

- **Awerbuch, Scheideler, Towards Scalable and Robust Overlay Networks**
- **Problem:**
  - Rejoin attacks
- **Solution:**
  - Chord network combined with
  - Cuckoo Hashing
  - Majority condition:
    - honest peers in the neighborhood are in the majority
  - Data is stored with O(log n) copies

Cuckoo Hashing

- **Collision strategy for (classical) hashing**
  - uses two hash functions h_1, h_2
  - an item with key x is either stored at $h_1(x)$ or $h_2(x)$
    - easy lookup
- **Insert x**
  - try inserting at $h_1(x)$ or $h_2(x)$
  - if both positions are occupied then
    - kick out one element
    - and insert it at its other place
    - continue this with the next element if the position is occupied

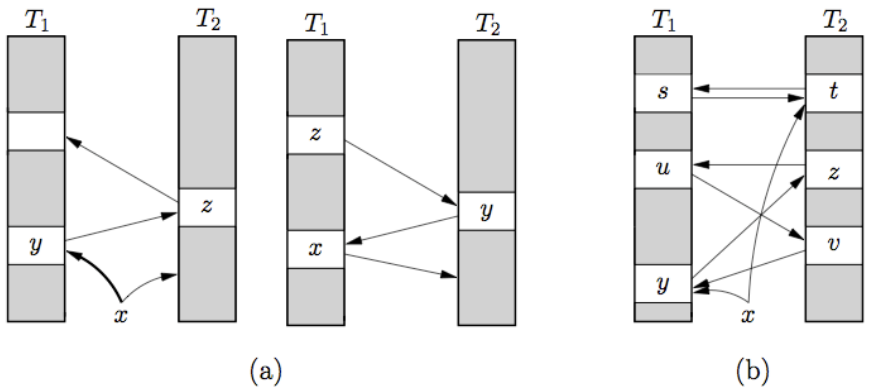


Fig. 1. Examples of CUCKOO HASHING insertion. Arrows show possibilities for moving keys. (a) Key x is successfully inserted by moving keys y and z from one table to the other. (b) Key x cannot be accommodated and a rehash is necessary.

From Cuckoo Hashing
Rasmus Pagh, Flemming Friche Rodler
2004

Efficiency of Cuckoo Hashing

- **Theorem**
  - Let $\epsilon>0$ then if at most n elements are stored, then Cuckoo Hashing needs a hash space of $2n+\epsilon$.
- **Three hash functions increase the load factor from 1/2 to 91%**
- **Insert**
  - needs O(1) steps in the expectation
  - O(log n) with high probability
- **Lookup**
  - needs two steps

Cuckoo Hashing for Security

- **Given n honest peers and ϵ n dishonest peers**
- **Goal**
  - For any adversarial attack the following properties for every interval $I \subseteq [0, 1)$ of size at least $(c \log n)/n$ we have
  - Balancing condition
    - I contains $\Theta(|I| \cdot n)$ nodes
  - Majority condition
    - the honest nodes in I are in the majority
- **Then all majority decisions of O(log n) nodes give a correct result**

Rejoin Attacks

- **Secure hash functions for positions in the Chord**
  - if one position is used
  - then in an O(log n) neighborhood more than half is honest
  - if more than half of al peers are honest
- **Rejoin attacks**
  - use a small number of attackers
  - check out new addresses until attackers fall in one interval
  - then this neighborhood can be ruled by the attackers

The Cuckoo Rule for Chord

- **Notation**
  - a region is an interval of size $1/2^r$ in [0, 1) for some integer r that starts at an integer multiple of $1/2^r$
  - There are exactly 2^r regions
  - A k-region is a region of size (closest from above to) k/n, and for any point $x \in [0, 1)$
  - the k-region $R_k(x)$ is the unique k-region containing x.
- **Cuckoo rule**
  - If a new node v wants to join the system, pick a random $x \in [0, 1)$.
  - Place v into x and move all nodes in $R_k(x)$ to points in [0, 1) chosen uniformly at random
    - (without replacing any further nodes).
- **Theorem**
  - For any constants ϵ and k with $\epsilon < 1-1/k$, the cuckoo rule with parameter k satisfies the balancing and majority conditions for a polynomial number of rounds, with high probability, for any adversarial strategy within our model.
  - The inequality $\epsilon < 1 - 1/k$ is sharp

Operations

- **Data storage**
  - each data item is stored in the $O(\log^3 n)$ neighborhood as copies
- **Primitives**
  - robust hash functions
    - safe against attacks
  - majority decisions of each operation
  - use multiple routes for targeting location

Efficiency

- **Lookup**
  - works correctly with high probability
  - can be performed with $O(\log^5 n)$ messages
- **Inserting of data**
  - works in polylogarithmic time
  - needs $O(\log^5 n)$ messages
- **Copies stored of each data: $O(\log^3 n)$**

Discussion

- **Advantage**
  - Cuckoo Chord is safe against adversarial attacks
  - Cuckoo rule is simple and effective
- **Disadvantage**
  - Computation of secure hash function is complex
  - Considerate overhead for communication
- **Theoretical breakthrough**
- **Little impact to the practical world**

Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

Peer-to-Peer Networks

NAT, PAT & Firewalls

Network Address Translation

- **Problem**
  - too few (e.g. one) IP addresses for too many hosts in a local network
  - hide hosts IP addresses from the outer world
- **Basic NAT (Static NAT)**
  - replace internal IP by an external IP
- **Hiding NAT**
  - = PAT (Port Address Translation)
  - = NAPT (Network Address Port Translation)
  - Socket pair (IP address and port number) are transformed
  - to a single outside IP address
- **Hosts in local network cannot be addressed from outside**

DHCP Dynamic Host Configuration Protocol

- **DHCP (Dynamic Host Configuration Protocol)**
  - manual binding of MAC address
    - e.g. for servers
  - automatic mapping
    - fixed, yet not pre-configured
  - dynamic mapping
    - addresses may be reused
- **Integration of new hosts without configuration**
  - hosts fetches IP address from DHCP server
  - sever assigns address dynamically
  - when the hosts leaves the network the IP address may be reused by other hosts
  - for dynamic mapping addresses must be refreshed
  - if a hosts tries to reuse an outdated address the DHCP server denies this request
  - problem: stealing of IP addresses
- **P2P**
  - DHCP is good for anonymity
    - if the DHCP is safe
  - DHCP is bad for contacting peers in local networks

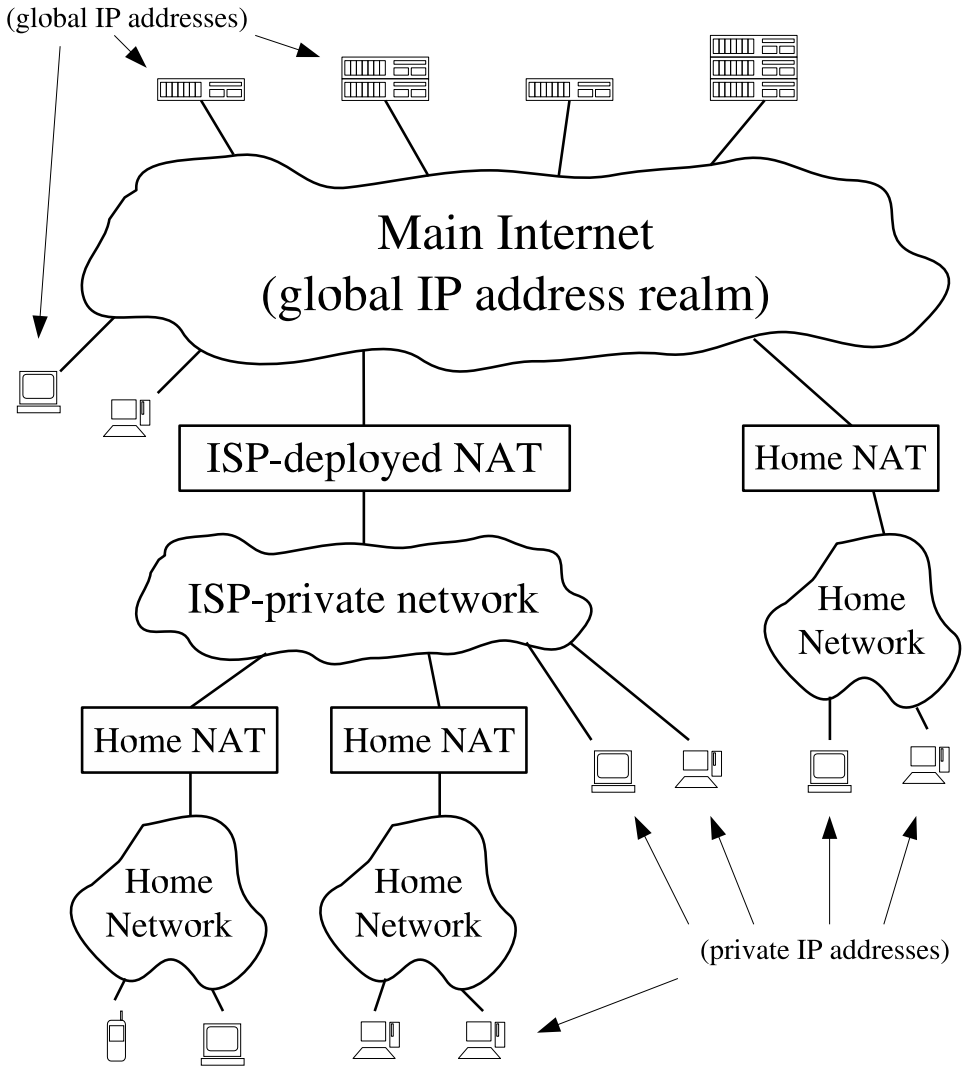
Firewalls

- **Types of Firewalls**
  - Host Firewall
  - Network Firewall
- **Network Firewall**
  - differentiates between
    - external net
      - Internet, hostile
    - internal net
      - LAN, trustworthy
    - demilitarized zone
      - servers reachable from the external net
- **Host Firewall**
  - e.g. personal firewall
  - controls the complete data traffic of a host
  - protection against attacks from outside and inside (trojans)
- **Methods**
  - Packet Filter
    - blocks ports and IP addresses
  - Content Filter
    - filters spam mails, viruses, ActiveX, JavaScript from html pages
  - Proxy
    - transparent (accessible and visible) hots
    - channels the communication and attacks to secured hosts
  - Stateful Inspection
    - observation of the state of a connection
- **Firewalls can prevent Peer to Peer connections**
  - on purpose or as a side effect
  - are treated here like NAT

Types of Firewalls & NATs (RFC 3489)

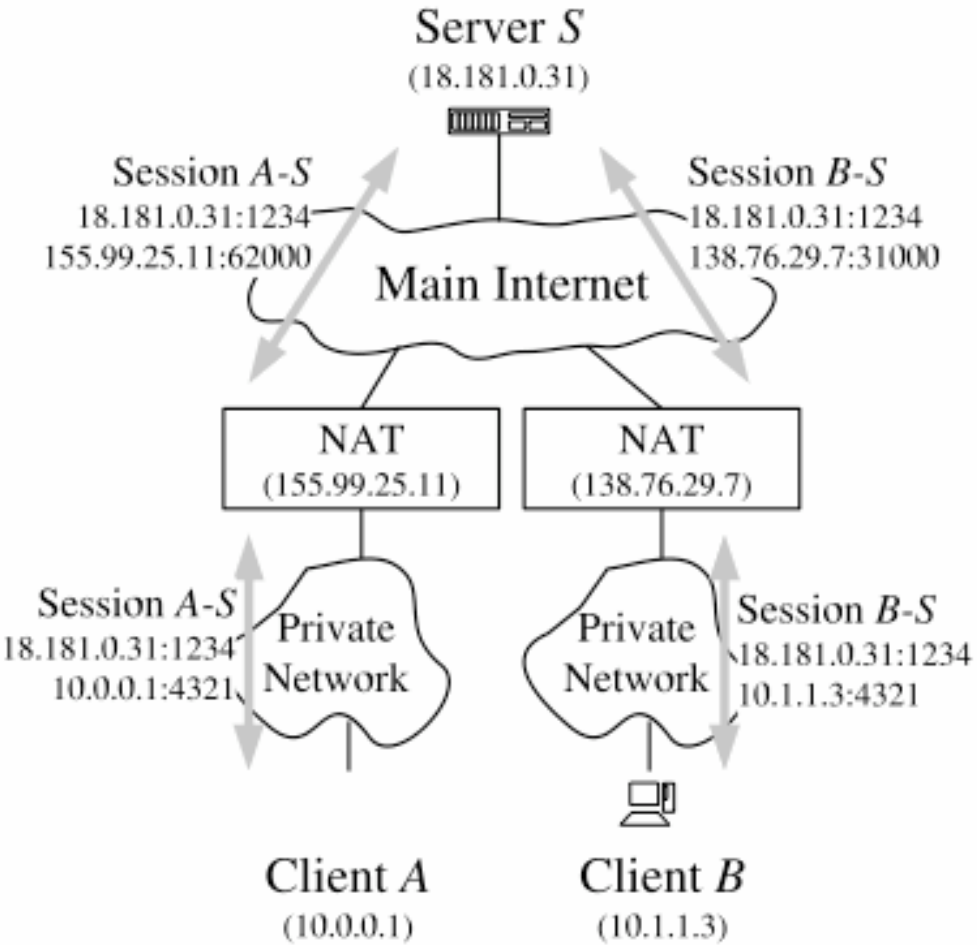
- **Open Internet**
  - addresses fully available
- **Firewall that blocks UDP**
  - no UDP traffic at all
  - hopeless, maybe TCP works?
- **Symmetric UDP Firewall**
  - allows UDP out
  - responses have to come back to the source of the request
  - like a symmetric NAT, but no translation
- **Full-cone NAT**
  - if an internal address is mapped to an external address all packets from will be sent through this address
  - External hosts can send packets to the external address which are delivered to the local address
- **Symmetric NAT**
  - Each internal request is mapped to a new port
  - Only a contacted host can send a message inside
    - on the very same external port arriving on the internal port
- **Restricted cone NAT**
  - Internal address are statically mapped to external addresses
  - All such UDP packets of one internal port use this external port
  - All external hosts can use this port to sent a packet to this host if they have received a packet recently from the same internal port (to any external port)
- **Port restricted cone NAT**
  - All UDP packets from one internal address use the same external port
  - External hosts must use this port to sent a packet to this host if they have received a packet recently from the same internal port to the same external port

Combination of NATs



Overcoming NAT by Relaying

- **Relaying**
  - use a open (non-NATed) server to relay all UDP or TCP connections
  - first both partners connect to the server
  - then, the server relays all messages

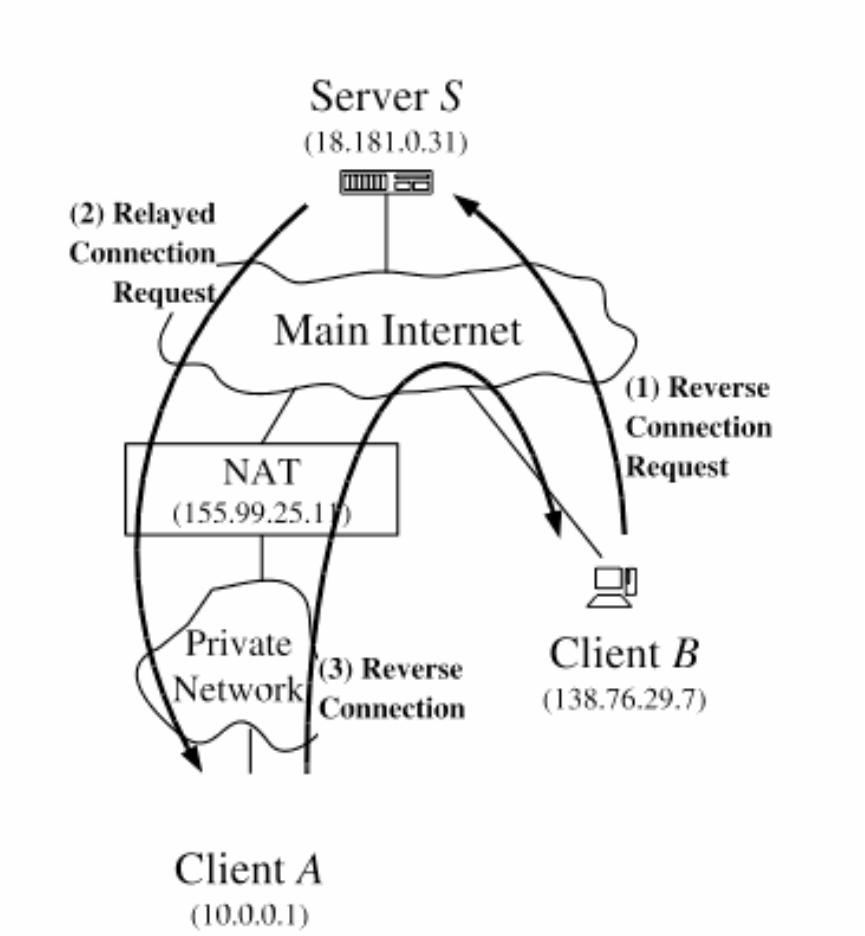


Peer-to-Peer Communication Accross Network Address Translators

Bryan Ford, Pyda Srisuresh, Dan Kegel

Connection Reversal

- **If only one peer is behind NAT**
  - then the peer behind NAT always starts connection
- **Use a server to announce a request for connection reversal**
  - periodic check for connection requests is necessary



Peer-to-Peer Communication Accross Network Address Translators

Bryan Ford, Pyda Srisuresh, Dan Kegel

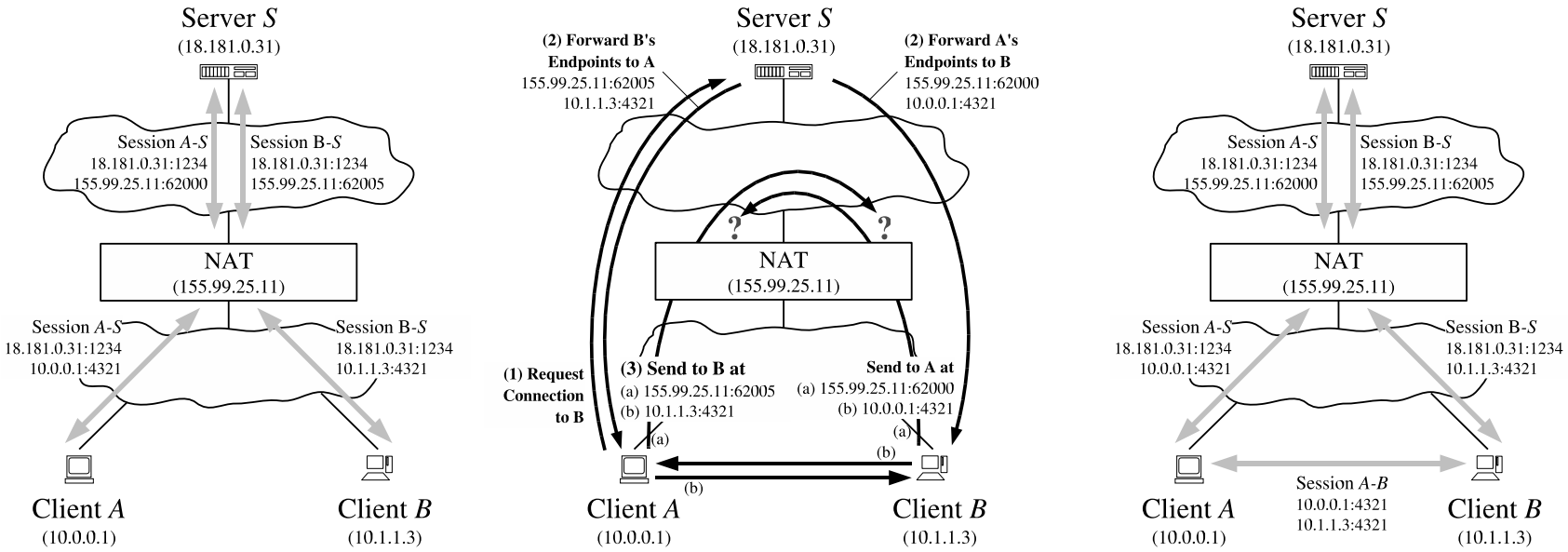
Peer-to-Peer Networks

UDP Hole Punching

UDP Hole Punching

- **Dan Kegel (1999), *NAT and Peer-to-Peer Networking*,** Technical Report Caltech
- **A does not know B‘s address**
- **Algorithm**
  - A contacts rendezvous server S and tells his local IP address
  - S replies to A with a message containing
    - B‘s public and private socket pairs
  - A sends UDP packets to both of this addresses
    - and stays at the address which works

UDP Hole Punching



Before Hole Punching

The Hole Punching Process

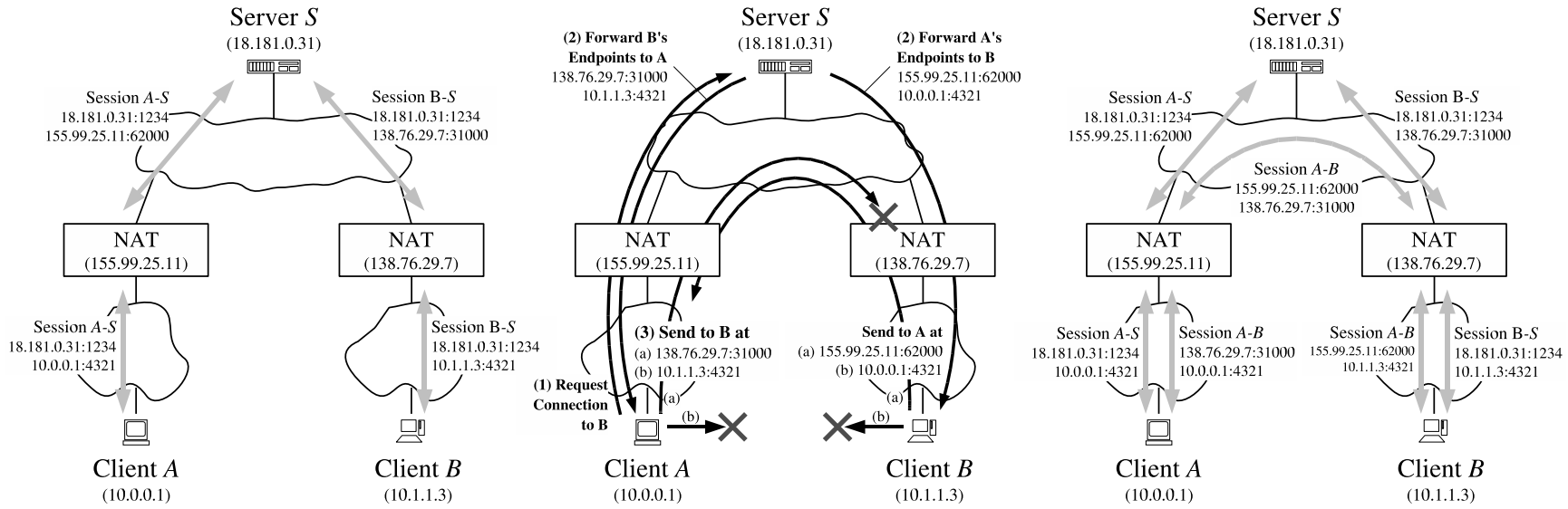
After Hole Punching

- **Peers Behind a Common NAT**
  - Rendezvous server is used to tell the local IP addresses
  - Test with local IP address establish the connections in the local net

Peer-to-Peer Communication Accross Network Address Translators

Bryan Ford, Pyda Srisuresh, Dan Kegel

UDP Hole Punching

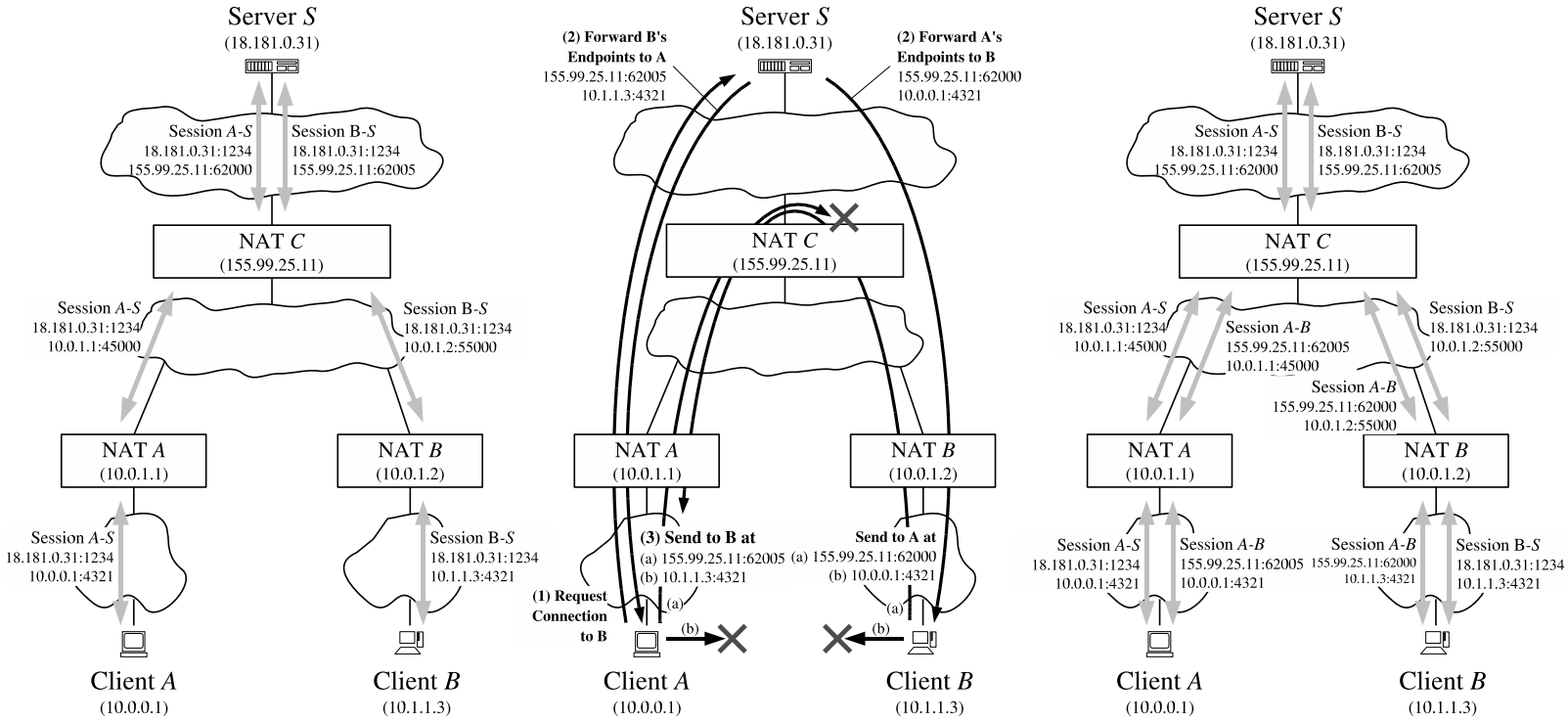


Before Hole Punching | The Hole Punching Process | After Hole Punching

- **Peers Behind Different NATs**
  - Rendezvous server is used to tell the NAT IP addresses
  - Test with NAT IP address establishes the connections
  - Peers reuse the port from the Rendezvous server

Peer-to-Peer Communication Accross Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

UDP Hole Punching



Before Hole Punching | The Hole Punching Process | After Hole Punching

- **Peers Behind Multiple Levels of NAT**
  - Rendezvous server is used to tell the NAT IP addresses
  - Test with NAT IP address establishes the connections
  - Relies on loopback translation of NAT C

Peer-to-Peer Communication Accross Network Address Translators

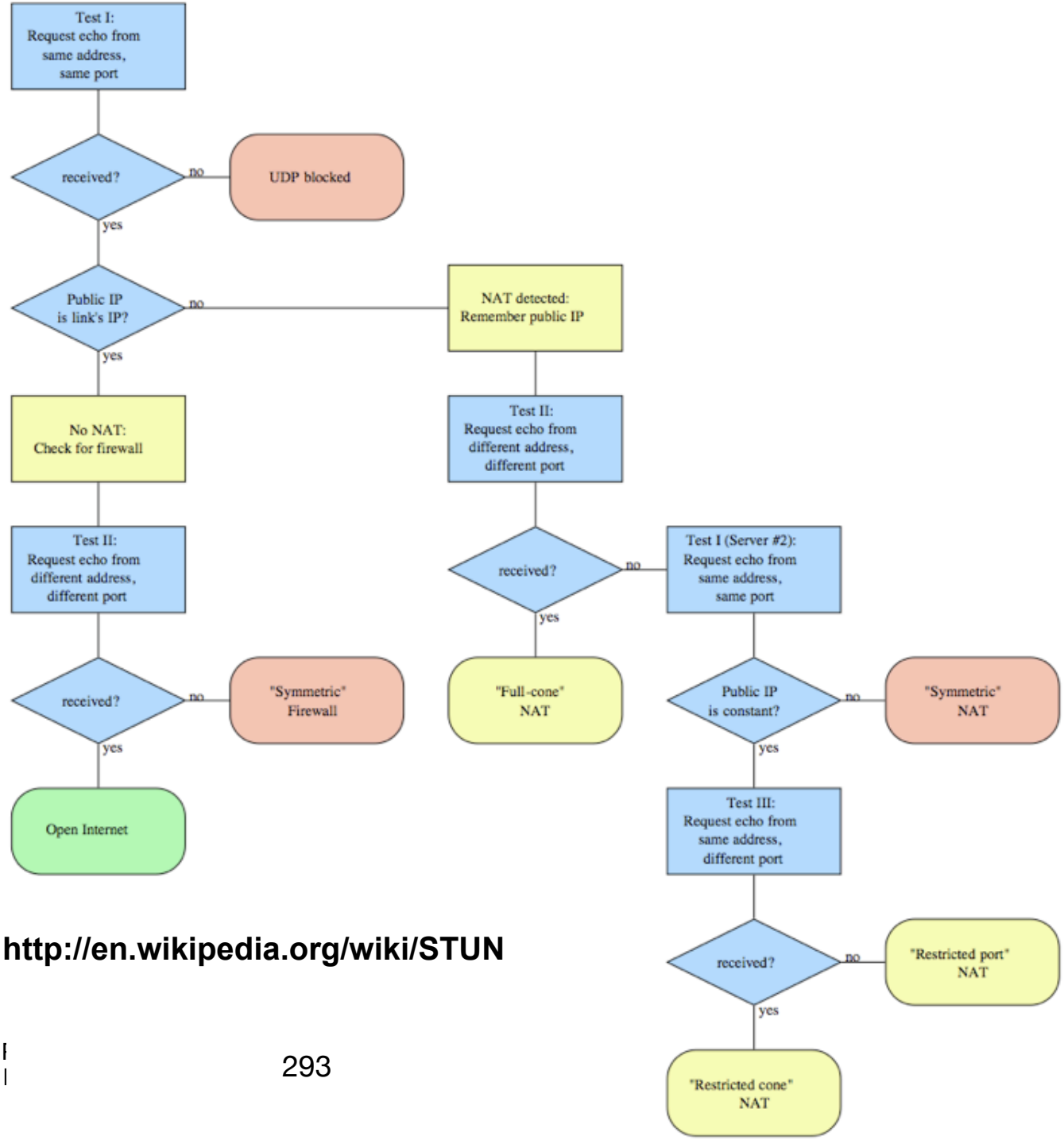
Bryan Ford, Pyda Srisuresh, Dan Kegel

Simple traversal of UDP over NATs (STUN)

- **RFC 3489, J. Rosenberg, C. Huitema, R. Mahy, STUN - Simple Traversal of User Datagram Protocol Through Network Address Translators (NATs), 2003**
- **Client-Server Protocol**
  - Uses open client to categorize the NAT router
- **UDP connection can be established with open client**
  - Tells both clients the external ports and one partner establishes the connection
- **Works for Full Cone, Restricted Cone and Port Restricted Cone**
  - Both clients behind NAT router can initialize the connection
  - The Rendezvous server has to transmit the external addresses
- **Does not work for Symmetric NATs**

STUN Test

- **Client communicates to at least two open STUN server**



NAT types

from: http://en.wikipedia.org/wiki/STUN

Peer-to-Peer Networks

TCP Hole Punching

TCP versus UDP Hole Punching

Category	UDP	TCP
Connection?	no	yes
Symmetry	yes	no client uses „connect“, server uses „accept“ or „listen“
Acknowledgments	no	yes must have the correct sequence numbers

P2P-NAT

Peer-to-Peer Communication Accross Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

- **Prerequisite**
  - change kernel to allow to listen and connect TCP connections at the same time
  - use a Rendezvous Server S
  - Client A and client B have TCP sessions with s
- **P2P-NAT**
  - Client A asks S about B‘s addresses
  - Server S tells client A and client B the public and private addresses (IP-address and port number) of A and B
  - From *the same local TCP ports* used to register with S
    - A and B synchronously make outgoing connection attempts to the others‘ public and private endpoints
  - A and B
    - wait for outgoing attempts to succeed
    - wait for incoming connections to appear
    - if one outgoing connection attempt fails („connection reset“, „host unreachable“) then the host retries after a short delay
  - Use the first established connection
  - When a TCP connection is made the hosts authenticate themselves

P2P-NAT

Peer-to-Peer Communication Accross Network Address Translators

Bryan Ford, Pyda Srisuresh, Dan Kegel

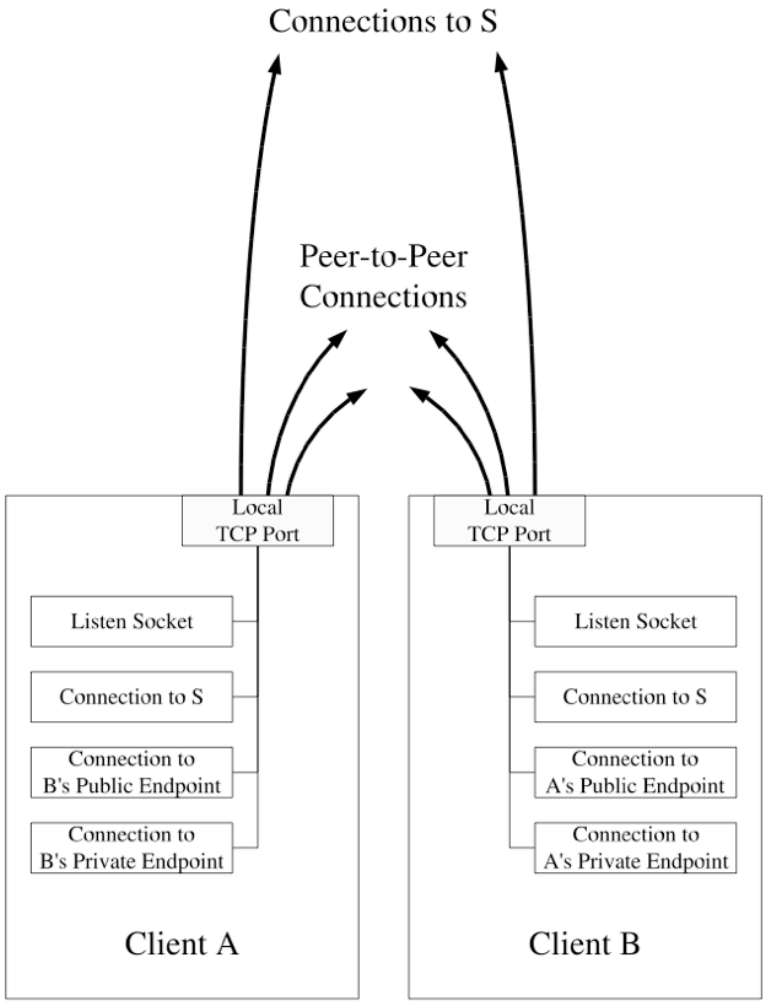


Figure 7: Sockets versus Ports for TCP Hole Punching

P2P-NAT

Peer-to-Peer Communication Accross Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

- **Behavior for *nice* NAT-routers of A**
  - The NAT router of A learns of outgoing TCP-connection when A contacts B using the public address
    - A has punched a hole in its NAT
  - A‘s first attempts may bounce from B‘s NAT router
  - B‘s connection attempt through A‘s NAT hole is successful
  - A is answering to B‘s connection attempt
  - B‘s NAT router thinks that the connection is a standard client server
- **Some packets will be dropped by the NAT routers in any case**
- **This connection attempt may also work if B has punched a hole in his NAT router before A**
  - The client with the weaker NAT router is the server in the TCP connection

P2P-Nat
Problems with Acks?

- **Suppose A has punched the hole in his router**

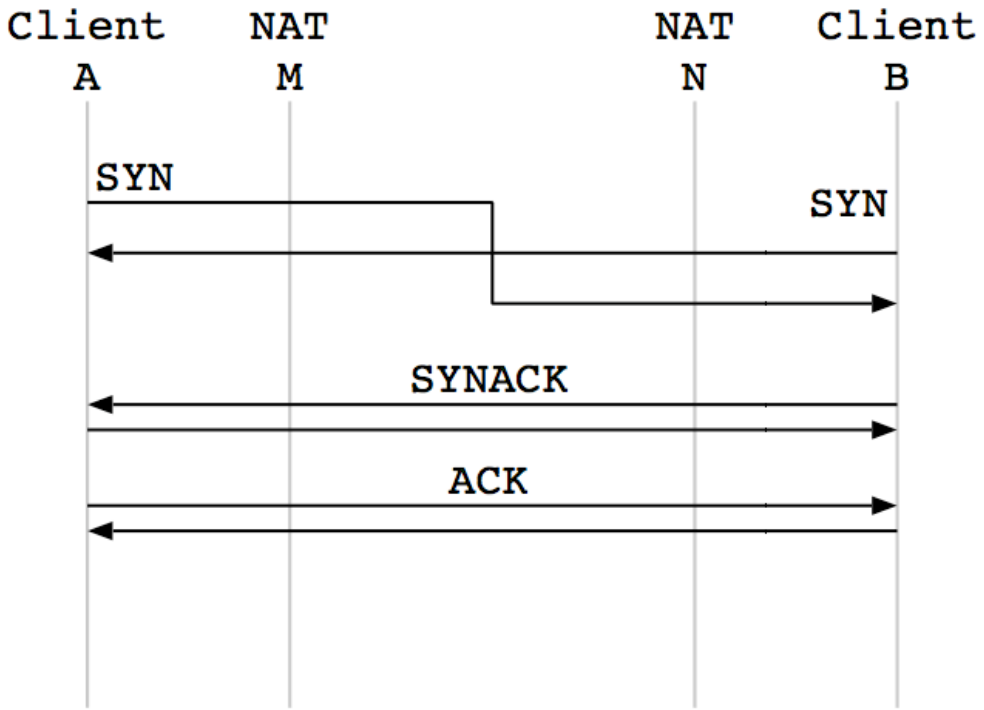
- **A sends SYN-packet**
- **but receives a SYN packet from B without Ack**
  - so the first SYN from A must be ignored
- **A replies with SYN-ACK to B**
- **B replies with ACK to A**
  - all is fine then

- **Alternatively:**
  - A might create a new stream socket associated with B‘s incoming connection start
    - a different stream socket from the socket that A hole punching TCP SYN message
    - this is regarded as a failed connection attempt
  - Also results in a working connection

P2P-NAT
The Lucky (?) Case

- **What if both clients A and B succeed synchronously?**
- **When both clients answere to the SYN with a SYN-ACK**
  - results in **simultaneous TCP open**
- **Can result in the failure of the connection**
  - depends on whether the TCP implementation accepts a simultaneous successful „accept()“ and „connect()“ operation
- **Then, the TCP connection should work correctly**
  - if the TCP implementation complies with RFC 793
- **The TCP connection has been „magically“ created itself from the wire**
  - out of nowhere two fitting SYN-ACKs have been created.

P2P-NAT Working Principle



(d) P2PNAT

Picture from
Characterization and Measurement of TCP Traversal through NATs and Firewalls
Saikat Guha, Paul Francis

Success Rate of UDP Hole Punching and P2P-NAT (2005)

	UDP				TCP			
	Hole Punching		Hairpin		Hole Punching		Hairpin	
NAT Hardware								
Linksys	45/46	(98%)	5/42	(12%)	33/38	(87%)	3/38	(8%)
Netgear	31/37	(84%)	3/35	(9%)	19/30	(63%)	0/30	(0%)
D-Link	16/21	(76%)	11/21	(52%)	9/19	(47%)	2/19	(11%)
Draytek	2/17	(12%)	3/12	(25%)	2/7	(29%)	0/7	(0%)
Belkin	14/14	(100%)	1/14	(7%)	11/11	(100%)	0/11	(0%)
Cisco	12/12	(100%)	3/9	(33%)	6/7	(86%)	2/7	(29%)
SMC	12/12	(100%)	3/10	(30%)	8/9	(89%)	2/9	(22%)
ZyXEL	7/9	(78%)	1/8	(13%)	0/7	(0%)	0/7	(0%)
3Com	7/7	(100%)	1/7	(14%)	5/6	(83%)	0/6	(0%)
OS-based NAT								
Windows	31/33	(94%)	11/32	(34%)	16/31	(52%)	28/31	(90%)
Linux	26/32	(81%)	3/25	(12%)	16/24	(67%)	2/24	(8%)
FreeBSD	7/9	(78%)	3/6	(50%)	2/3	(67%)	1/1	(100%)
All Vendors	310/380	(82%)	80/335	(24%)	184/286	(64%)	37/286	(13%)

Table 1: User Reports of NAT Support for UDP and TCP Hole Punching

Peer-to-Peer Communication Accross Network Address Translators
Bryan Ford, Pyda Srisuresh, Dan Kegel

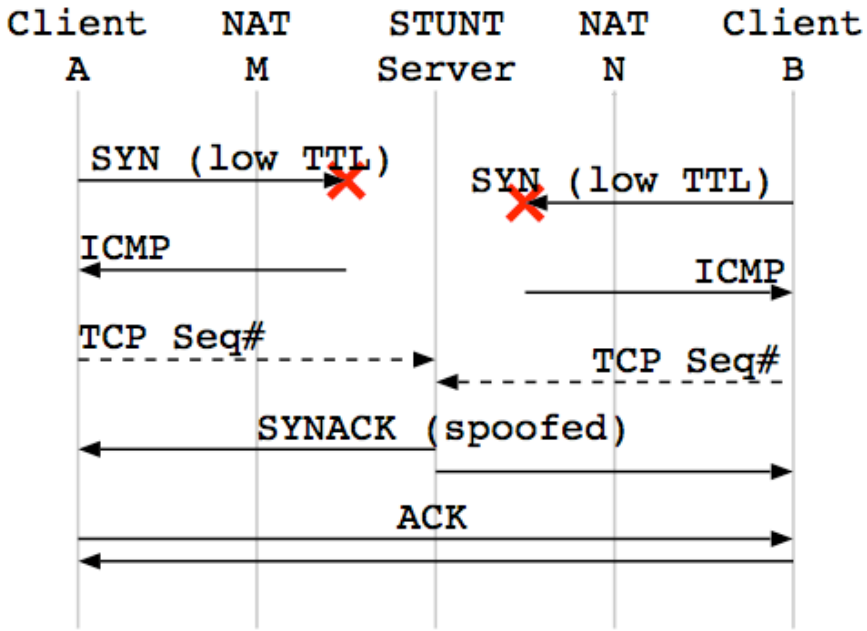
TCP Hole Punching with Small TTL

- **NAT Servers can be punched with TCP Sync packets of small TTL**
  - message passes NAT server
  - listening to outgoing messages help to learn the Sequence Number
- **Technique used by**
  - STUNT#1, #2
  - NATBlaster

STUNT

- **Both endpoints produce a SYN packet with small TTL**
  - Packet passes NAT-router, yet does not reach target
- **Both clients learn their own (!) sequence number**
- **STUNT (Rendezvous) server produces a spoofed SYNACK**
  - with correct sequence number to both clients
- **Both clients respond with ACK**
- **Hopefully, connection is established**
- **Problems:**
  - Choice of TTL. Not possible if the two outermost NATs share an interface
  - ICMP-packet can be interpreted as fatal error
  - NAT may change the sequence number, spoofed SYNACK might be „out of window“
  - Third-party spoofer is necessary

Eppinger, TCP Connections for P2P Apps: A Software Approach to Solving the NAT Problem. Tech. Rep. CMU-ISRI-05-104, Carnegie Mellon University, Pittsburgh, PA, Jan. 2005.

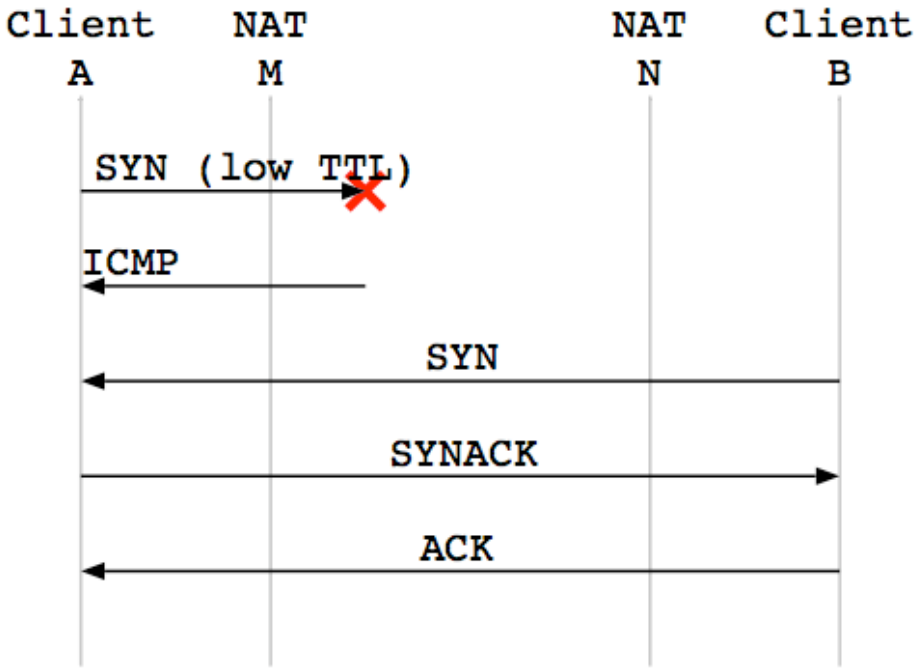


(a) STUNT #1

STUNT (version 2)

- **Endpoints A produce a SYN packet with small TTL**
  - Packet passes NAT-router, yet does not reach target
- **Client A aborts attemption connect**
  - accepts inbound connections
- **Client B**
  - learns address from Rendezvous server
  - initiates regular connection to A
- **Client A answers with SYNACK**
  - Hopefully, connection is established
- **Problems:**
  - Choice of TTL.
  - ICMP-packet must be interpreted as fatal error or
  - NAT must accept an inbound SYN following an outbound SYN
    - unusual situation

Guha, Takeda, Francis, NUTSS: A SIP-based Approach to UDP and TCP Network Connectivity. In Proceedings of SIGCOMM'04 Workshops (Portland, OR, Aug. 2004), pp. 43– 48.

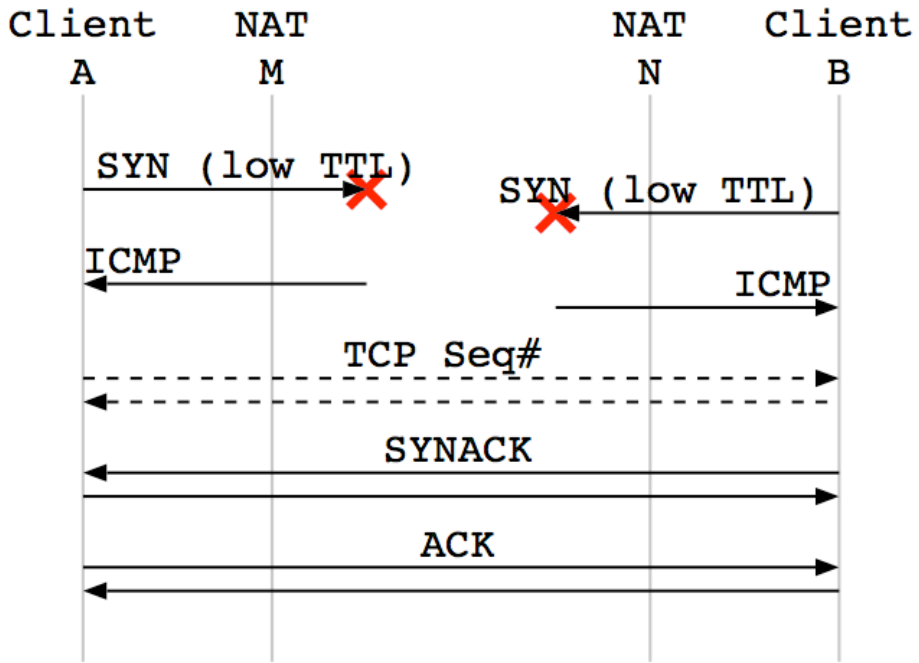


(b) STUNT #2

NATBlaster

- **Both endpoints produce low TTL SYN-packets**
  - passes NAT router, but does not reach other NAT router
- **Learn sequence number for own connection**
  - exchange this information using Rendezvous server
- **Both endpoints produce SYN-ACK packets**
  - Both endpoints answer with ACKs
  - Connection established
- **Problems**
  - Choice of TTL
  - NATs must ignore ICMP-packet
  - NAT may change sequence numbers
  - NAT must allow symmetric SYN-Acks after own SYN packet
    - unusual

Biggadie, Ferullo, Wilson, Perrig, NATBLASTER: Establishing TCP connections between hosts behind NATs. In Proceedings of ACM SIGCOMM, ASIA Workshop (Beijing, China, Apr. 2005).



(c) NATBlaster

OS Issues of TCP Hole Punching

Approach	NAT/Network Issues	Linux Issues	Windows Issues
STUNT #1	• Determining TTL • ICMP error • TCP Seq# changes • IP Address Spoofi ng	• Superuser priv.	• Superuser priv. • Setting TTL
STUNT #2	• Determining TTL • ICMP error • SYN-out SYN-in		• Setting TTL
NATBlaster	• Determining TTL • ICMP error • TCP Seq# changes • SYN-out SYNACK-out	• Superuser priv.	• Superuser priv. • Setting TTL • RAW sockets (post WinXP SP2)
P2PNAT	• TCP simultaneous open • Packet fbod		• TCP simultaneous open (pre WinXP SP2)
STUNT #1 no-TTL	• RST error • TCP Seq# changes • Spoofi ng	• Superuser priv.	• Superuser priv. • TCP simultaneous open (pre WinXP SP2)
STUNT #2 no-TTL	• RST error • SYN-out SYN-in		
NATBlaster no-TTL	• RST error • TCP Seq# changes • SYN-out SYNACK-out	• Superuser priv.	• Superuser priv. • RAW sockets (post WinXP SP2) • TCP simultaneous open (pre WinXP SP2)

from Characterization and Measurement of TCP Traversal through NATs and Firewalls, Saikat Guha, Paul Francis

Port Prediction

- **NAT router changes port addresses for incoming connections**
- **A knows the type of NAT**
  - learns the mapping from the Rendezvous (STUNT) server
  - predicts its mapping
- **B also predicts his mapping**
- **Both clients send SYN packets to the predicted ports**
- **Usually, NAT servers can be very well predicted, e.g.**
  - outgoing port is 4901.
  - then the incoming port is 4902
    - if 4902 is not used, then it is 4903
      - and so on....

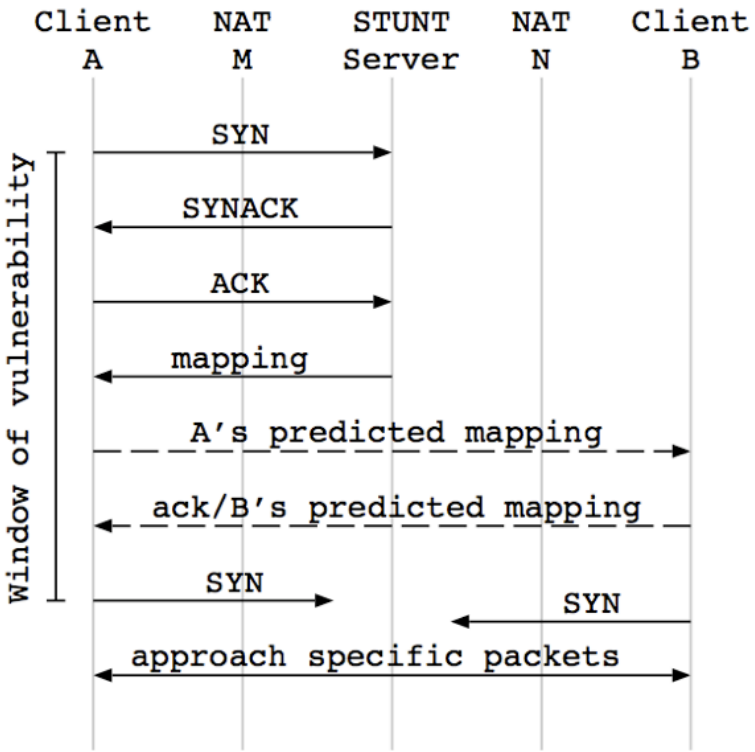


Figure 6: Port-prediction in TCP NAT-Traversal approaches. **from** Characterization and Measurement of TCP Traversal through NATs and Firewalls, Saikat Guha, Paul Francis

How Skype Punches Holes

- **An Experimental Study of the Skype Peer-to-Peer VoIP System, Saikat Guha, Neil Daswani, Ravi Jain**
  - Skype does not publish its technique
  - Yet, behavior can be easily tracked
- **Techniques**
  - Rendezvous Server
  - UDP Hole Punching
  - Port scans/prediction
  - Fallback: UDP Relay Server
    - success rate of Skype very high, seldomly used

Overview

- **Motivation & short history**
  - Motivation
  - Short history
- **P2P Algorithms**
  - Distributed Hash Tables
  - Structured networks
- **P2P Tricks**
  - Self-organization
  - Game theory
  - Network Coding
- **P2P Problems**
  - Anonymity & Security
  - Internet

ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

CoNe
Freiburg

Algorithms and Methods for Peer-to-Peer Networks

End of Tutorial

Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Computer Networks and Telematics
Christian Schindelhauer