Peer-to-Peer Systems

Winter semester 2014

Jun.-Prof. Dr.-Ing. Kalman Graffi

Heinrich Heine University Düsseldorf

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems - Chapter 2

Unstructured P2P Overlays

- Main Search Principles
- Centralized Overlay
- Gnutella 0.4
- BubbleStorm

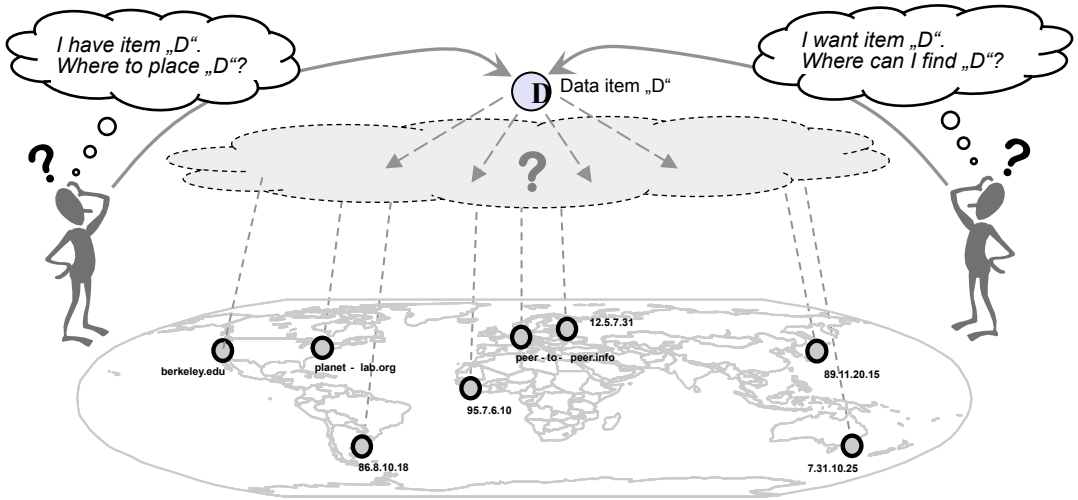
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Classification of P2P Overlay Networks
– Design Concepts

This slide set is based on the lecture "Communication Networks 2" of Prof. Dr.-Ing. Ralf Steinmetz at TU Darmstadt

Challenges in P2P related Data Management and Retrieval



Essential challenges in (most) Peer-to-Peer systems

- Location of a data items at distributed systems
  - Where shall the item be stored by the provider?
  - How does a requester find the actual location of an item?

Challenges

- Scale – potentially millions of nodes
- Dynamism – nodes go regularly online and offline

Major Query Types in P2P Overlays

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Lookup

- Given: ID (of object / node)
- Return: contact to responsible peer

Pros

- Each object uniquely identifiable
- Protocols guarantee results, if target exists
- Object location can be made efficient

Cons

- Need to know unique name
- Need to maintain address structure

(Full Text) Search

- Given: sequence of word (search term)
- Return: contact to peers with all resources matching the search terms

Pros

- No need to know unique names

Cons

- Protocols require exhaustive search in worst case
- Hard to make efficient

Structured and Unstructured P2P Networks

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Unstructured P2P Networks

- objects have no special identifier
- location of desired object a priori not known
- each peer is only responsible for objects it submitted

Structured P2P Networks

- peers and objects have identifiers
- objects are stored on peers according to their ID:
  responsibleFor(ObjID) = PeerID
- distributed indexing points to object location

Search:

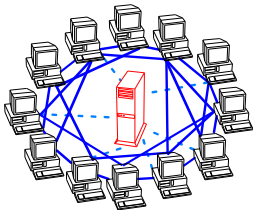
- Find all (or some) objects in the P2P network which fit to given criteria

Lookup / Addressing:

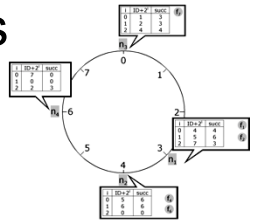
- Retrieve the object which is identified with a given identifier

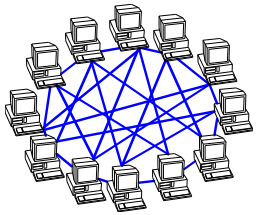
Types of P2P Overlays

Centralized

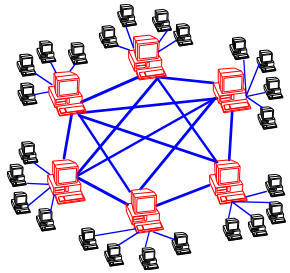


- Server is used to search for file owner
- Later: direct P2P communication is established
- Example: Napster, centralized DHT also pos





Homogeneous



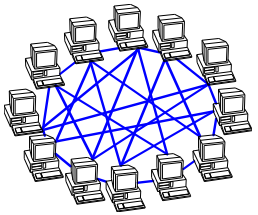
- All peers have equal duties, homogenous roles
- Example: Chord, Kademlia, Gnutella

Heterogeneous

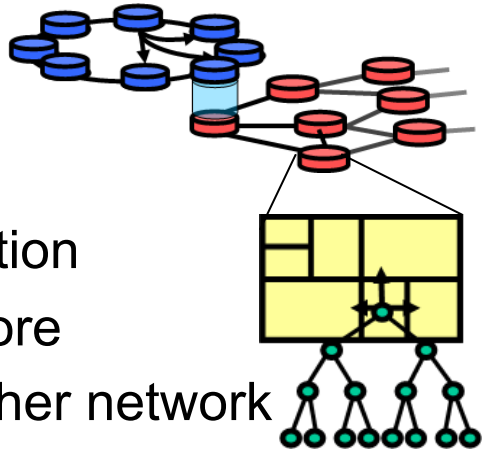
- Peers have different duties, heterogeneous roles
- Example, unstructured P2P overlays: KaZaA, eDonkey
- Example, structured P2P overlays: Adaptive-Chord

Classification of P2P Overlays

Flat



- All nodes are in only one overlay
- All functions performed in that single overlay
- Example: most overlays (Chord, Kademlia, Gnutella)



Hierarchical

- Several overlays exist, each with dedicated function
- Some nodes are in only one overlay, some in more
- Routing in one network leads to gateway in another network
  - Overall overlay is clustered
  - Inter-cluster routing, sometimes different ID space
- Example: like Border Gateway Routing

Metrics, Searching and Addressing

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Probability of success

- structured
  - protocols guarantee results, if target exists
    - assuming absence of malicious peers
- unstructured
  - protocols require exhaustive search

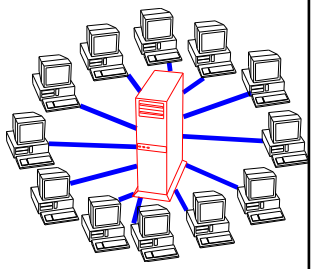
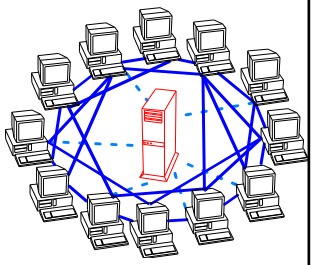
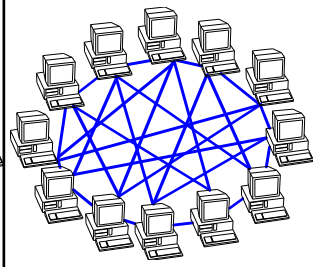
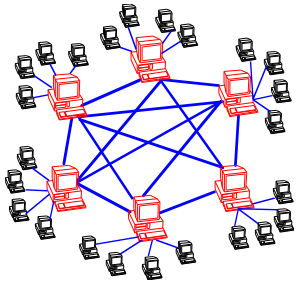
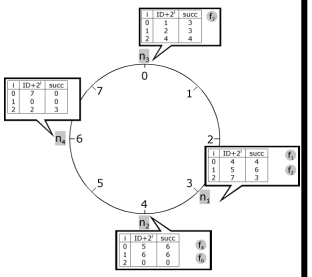
Protocol metrics

- Average number of messages per node
- Visited nodes
- Peak number of messages
- Congestion

Quality of results

- Completeness (are all results returned)
- Correctness (are all returned entries valid)
- Operation latency (time needed to solve the query)

Overlay Networks: Structures

Client-Server	***Peer-to-Peer***			
1. Server is the central entity and only provider of service and content. → Network managed by the Server 2. Server as the higher performance system. 3. Clients as the lower performance system Example: WWW	1. Resources are shared between the peers 2. Resources can be accessed directly from other peers 3. Peer is provider and requestor (Servent concept)			
	Unstructured P2P			***Structured P2P***
	Centralized	*Homogeneous*	*Heterogeneous*	*DHT-Based*
	1. All features of Peer-to-Peer included 2. Central entity is necessary to provide the service 3. Central entity is some kind of index/group database Example: Napster	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → no central entities Example: Gnutella 0.4, Freenet	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → dynamic central entities Examples: Gnutella 0.6, Fasttrack, edonkey	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities 4. Connections in the overlay are “fixed” 5. Distributed indexing (content is not relocated) Examples: Chord, CAN
				

from R.Schollmeier and J.Eberspächer, TU München

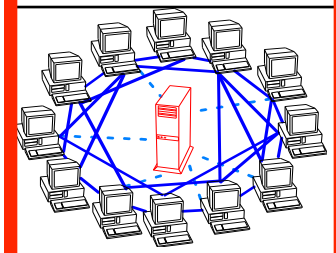
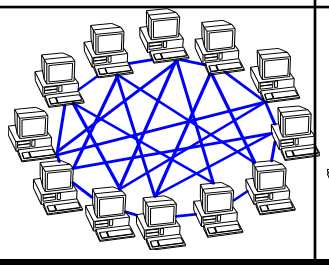
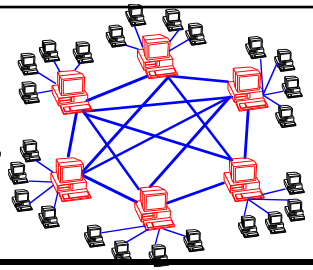
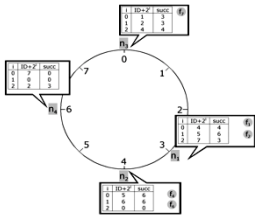
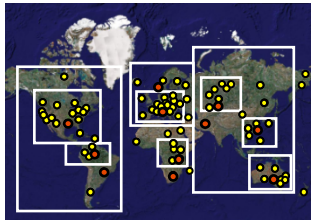
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Unstructured P2P Overlay Networks
– Centralized Approach

This slide set is based on the lecture "Communication Networks 2" of Prof. Dr.-Ing. Ralf Steinmetz at TU Darmstadt

Unstructured Centralized P2P Overlays

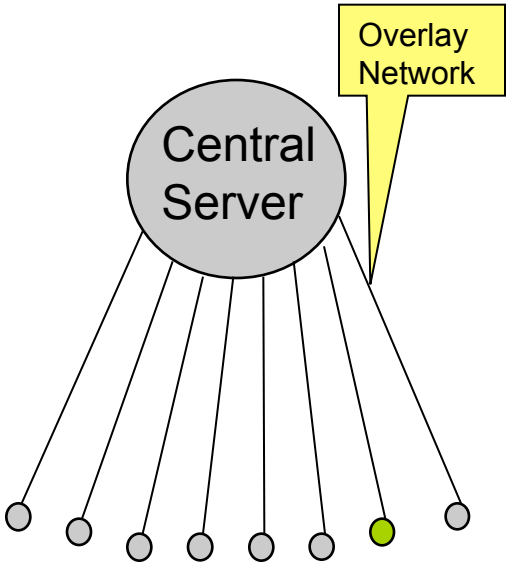
Unstructured P2P			*Structured P2P*	
Centralized P2P	***Homogenous P2P***	***Heterogeneous P2P***	***DHT-Based***	***Heterogeneous P2P***
1. All features of Peer-to-Peer included **2. Central entity is necessary to provide the service** **3. Central entity is some kind of index/group database** **Examples:** ▪ **Napster**	**1. All features of Peer-to-Peer included** **2. Any terminal entity can be removed without loss of functionality** **3. → no central entities** **Examples:** ▪ **Gnutella 0.4** ▪ **Freenet**	**1. All features of Peer-to-Peer included** **2. Any terminal entity can be removed without loss of functionality** **3. → dynamic central entities** **Examples:** ▪ **Gnutella 0.6** ▪ **Fasttrack** ▪ **eDonkey**	**1. All features of Peer-to-Peer included** **2. Any terminal entity can be removed without loss of functionality** **3. → No central entities** **4. Connections in the overlay are "fixed"** **Examples:** ▪ **Chord** ▪ **CAN** ▪ **Kademlia**	**1. All features of Peer-to-Peer included** **2. Peers are organized in a hierarchical manner** **3. Any terminal entity can be removed without loss of functionality** **Examples:** • **AH-Chord** • **Globase.KOM**
				

from R.Schollmeier and J.Eberspächer, TU München

Centralized P2P Networks

Central index server, maintaining index:

- What:
  - object name, file name, criteria (ID3) …
- Where
  - (IP address, Port)
- Search engine
  - Combining object and location information
- Global view on the network



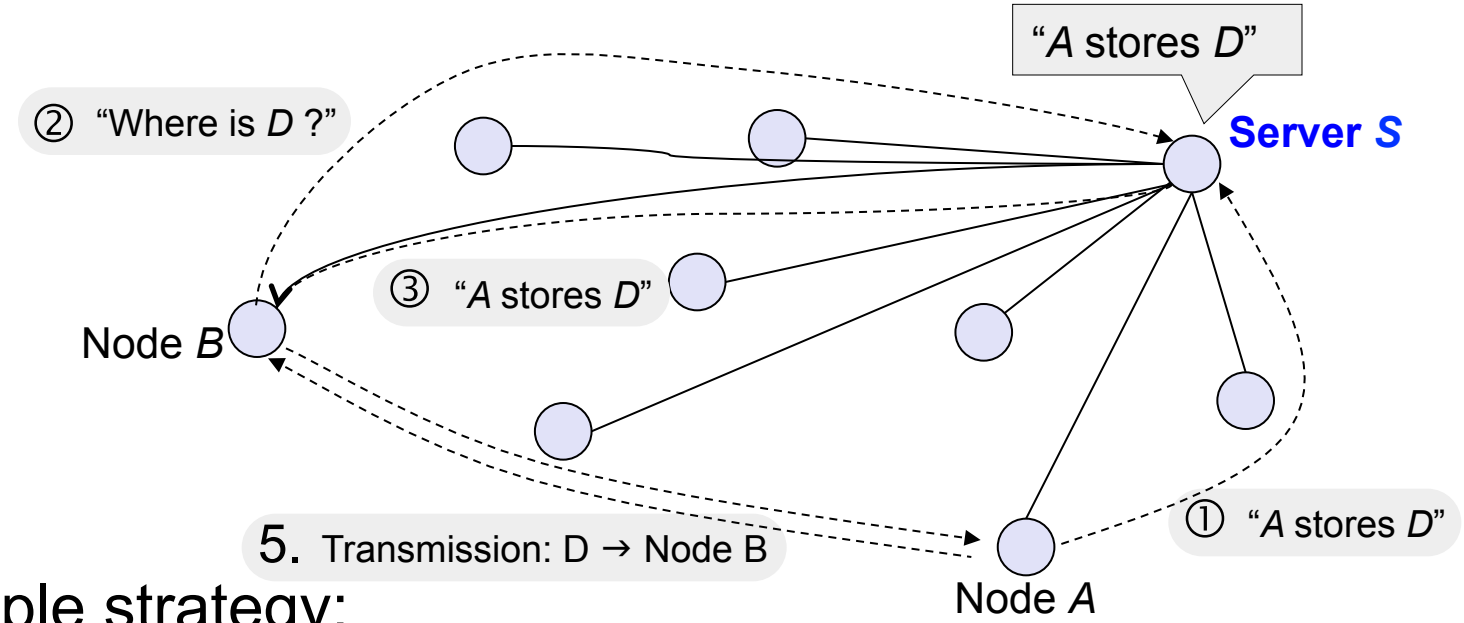
Normal peer, maintaining the objects:

- Each peer maintains only its own objects
- Decentralized storage (content at the edges)
- File transfer between clients (decentralized)

Issues:

- Unbalanced costs: central server is bottleneck
- Security: server is single point of failure

Centralized P2P Network



Simple strategy:

- Central server stores information about locations
  - 1. Node A (provider) tells server that it stores item D
  - 2. Node B (requester) asks server S for location of D
  - 3. Server S tells B that node A stores item D
  - (4. Node B checks the quality/bandwidth of the item providers)
  - 5. Node B requests item D from node A

Approach I: Central Server

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Advantages

- Search complexity of O(1) – “just ask the server”
- Lookup and search possible
- Simple and fast

Challenges

- No intrinsic scalability
  - O(N) network and system load of server
- Single point of failure or attack
- Non-linear increasing implementation and maintenance cost
  - in particular for achieving high availability and scalability
- Central server not suitable for systems with massive numbers of users

But overall, …

- Best principle for small and simple applications

Centralized P2P Networks (like Napster, ICQ)

Napster - History:

1999-2001:

- first famous P2P file sharing tool for mp3 files
- free content access, violation of copyright of artists

2001-2003:

- by introduction of filters
  - → users abandoned service
- April 2001: OpenNap
  - appr. 45.000 users,
  - 80 (interconnected) directory servers,
  - more than 50 TB data

2004: Napster 2

- music store
- no P2P network anymore
- download music tracks with a subscription model

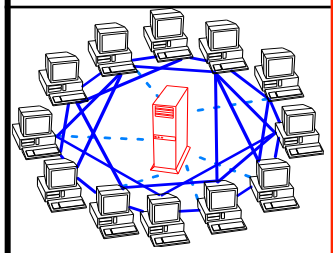
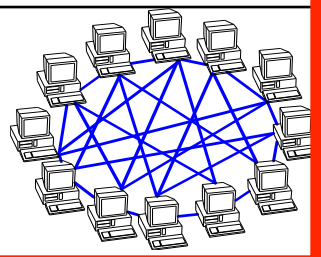
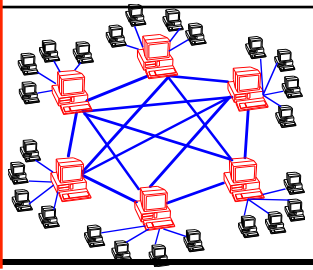
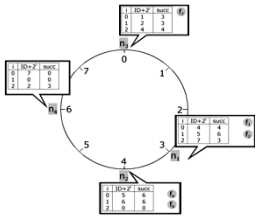
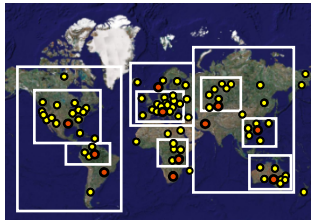
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Unstructured P2P Overlay Networks
– Search Principles

This slide set is based on the lecture "Communication Networks 2" of Prof. Dr.-Ing. Ralf Steinmetz at TU Darmstadt

Unstructured Homogenous P2P Overlays

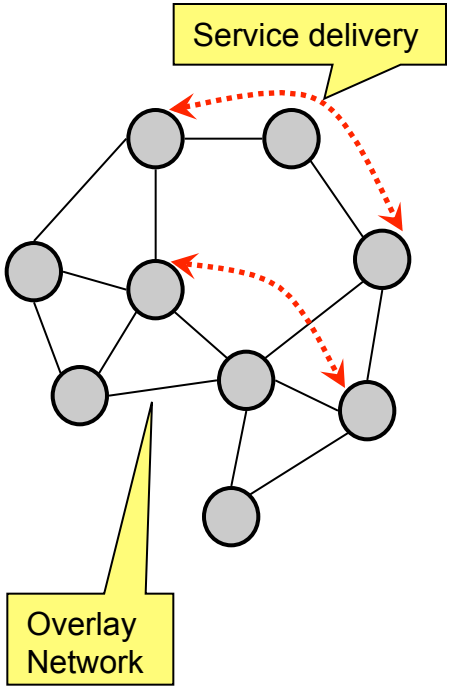
Unstructured P2P			*Structured P2P*	
Centralized P2P	***Homogeneous P2P***	***Heterogeneous P2P***	***DHT-Based***	***Heterogeneous P2P***
1. All features of Peer-to-Peer included **2. Central entity is necessary to provide the service** **3. Central entity is some kind of index/group database** **Examples:** ▪ **Napster**	**1. All features of Peer-to-Peer included** **2. Any terminal entity can be removed without loss of functionality** **3. → no central entities** **Examples:** ▪ **Gnutella 0.4** ▪ **Freenet**	**1. All features of Peer-to-Peer included** **2. Any terminal entity can be removed without loss of functionality** **3. → dynamic central entities** **Examples:** ▪ **Gnutella 0.6** ▪ **Fasttrack** ▪ **eDonkey**	**1. All features of Peer-to-Peer included** **2. Any terminal entity can be removed without loss of functionality** **3. → No central entities** **4. Connections in the overlay are "fixed"** **Examples:** ▪ **Chord** ▪ **CAN** ▪ **Kademlia**	**1. All features of Peer-to-Peer included** **2. Peers are organized in a hierarchical manner** **3. Any terminal entity can be removed without loss of functionality** **Examples:** • **AH-Chord** • **Globase.KOM**
				

from R.Schollmeier and J.Eberspächer, TU München

Distributed / Homogeneous P2P Overlays

Characteristics

- All peers are equal
  - (in their roles)
- Search mechanism is provided by cooperation of all peers
- Local view on the network



Organic growing:
Just append to current network

- No special infrastructure element needed

Motivation:

- To provide robustness
- To have self organization

Tasks to solve

1. To connect to the network
   - No central index server → Joining strategies needed
   - To join → to know at least 1 peer of the network
   - Local view on network → advertisements needed

2. To search
   - Search Mechanisms in P2P Overlays

3. To deliver the service
   - Establish connection to other node(s)
   - Peer to peer communication

Brainstorming for search mechanisms

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Identify various search mechanisms for random networks

Tradeoff:

- Message overhead
- (Un)completeness of search results

Tasks:

- Inserting new file
- Searching for file

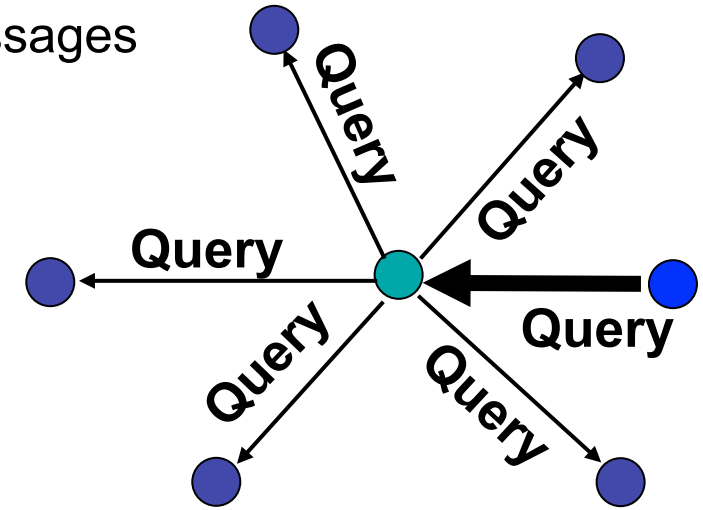
Search Mechanisms: Flooding

Unstructured p2p overlay

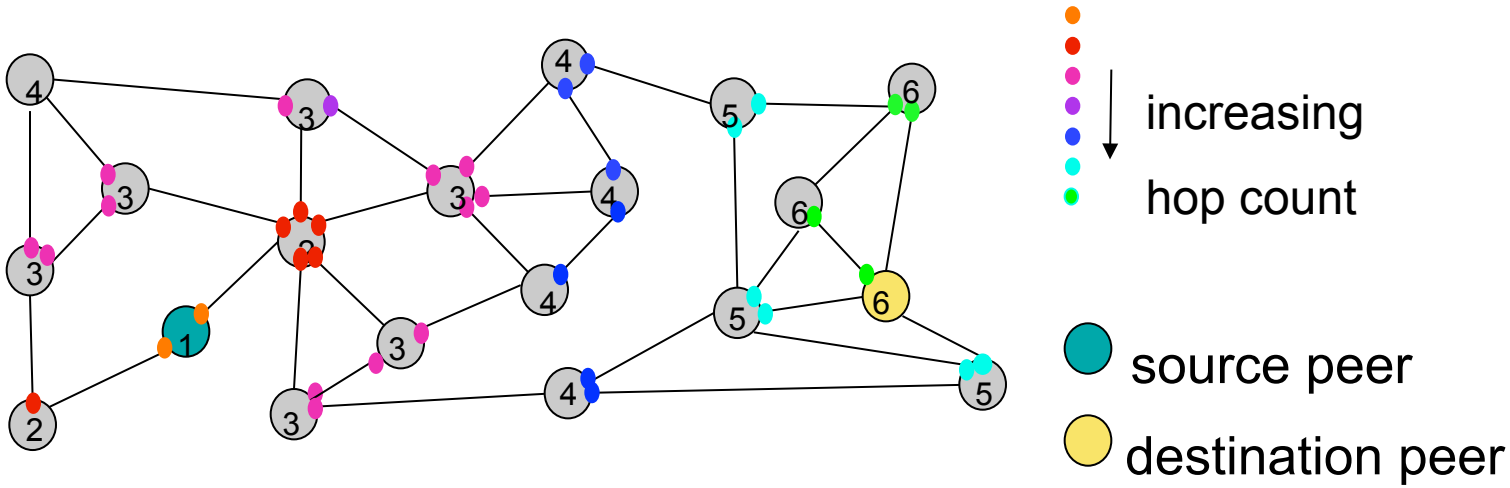
- No global information available about location of a item
- Content only stored at respective node providing it

Breadth-first search (BFS)

- Send a message to all neighbors
  - except the one who delivered the incoming message
- Use system-wide maximum Time-to-Live (TTL)
  - Decrement with every hop
  - Discard message if TTL negative
- Store message identifiers of routed messages to avoid retransmission cycles
- Highest effort for searching



Example



Overhead

- Large, here 43 messages sent

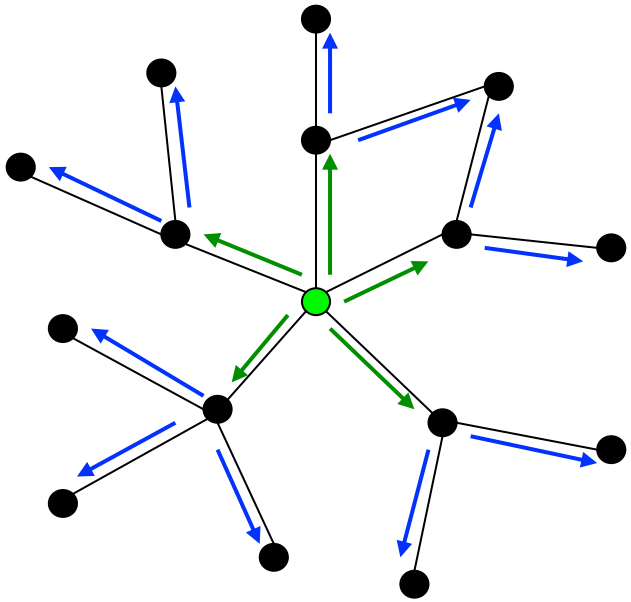
Length of the path:

- 5 hops

Search Mechanisms: Expanding Ring

Mechanism

- Successive floods with increasing TTL
  - Start with small TTL
  - If no success increase TTL
  - Example: 3-5-7-…



Properties

- Improved performance
  - if objects follow Zipf law popularity distribution and are located accordingly
- Message overhead is high

Excursion: Zipf – Distribution

Describes the probability of an element to occur

- N be the number of elements
  - E.g. 1000 elements
- k be their rank
  - E.g. element i is the 623th most frequent one to occur
- s be the value of the exponent characterizing the distribution

Frequency of an element k to occur:

Rank ^ (-s)

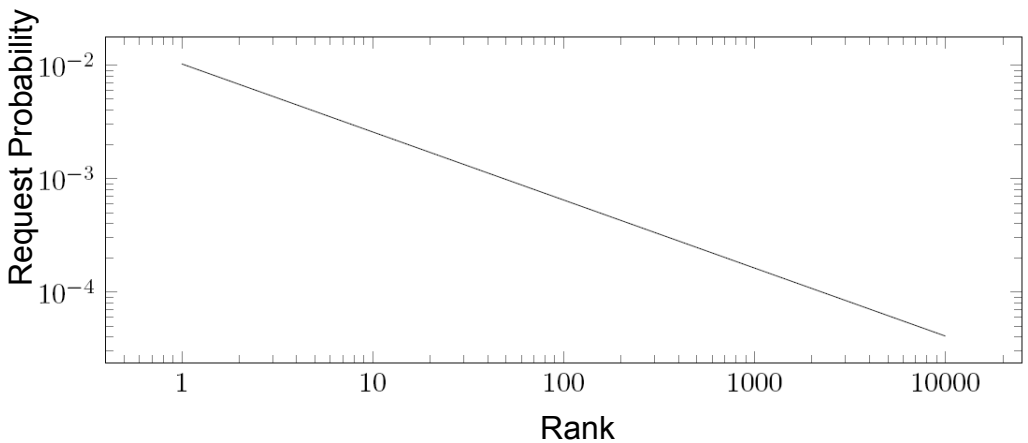
$$f(k;s,N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s}$$

Sum of all probabilities

Simple example:

- N =10000, s= 0.6

$$P(x) \sim \left(\frac{1}{rank(x)}\right)^{0.6}$$

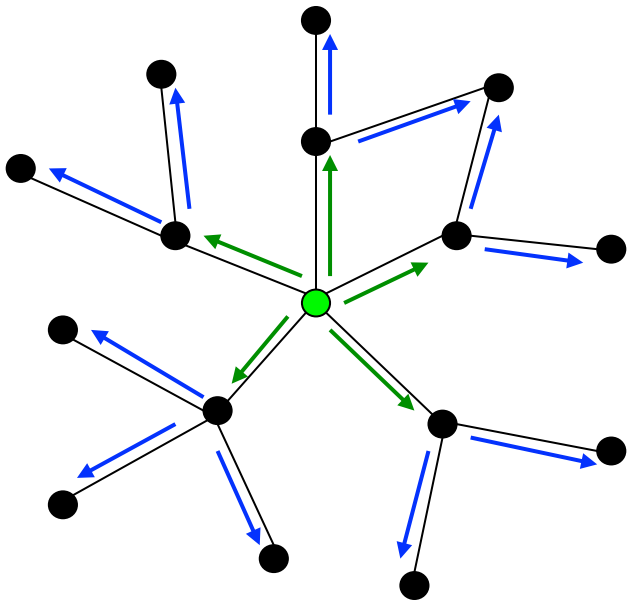


How to use this?

Search Mechanisms: Random Walk

Random walks

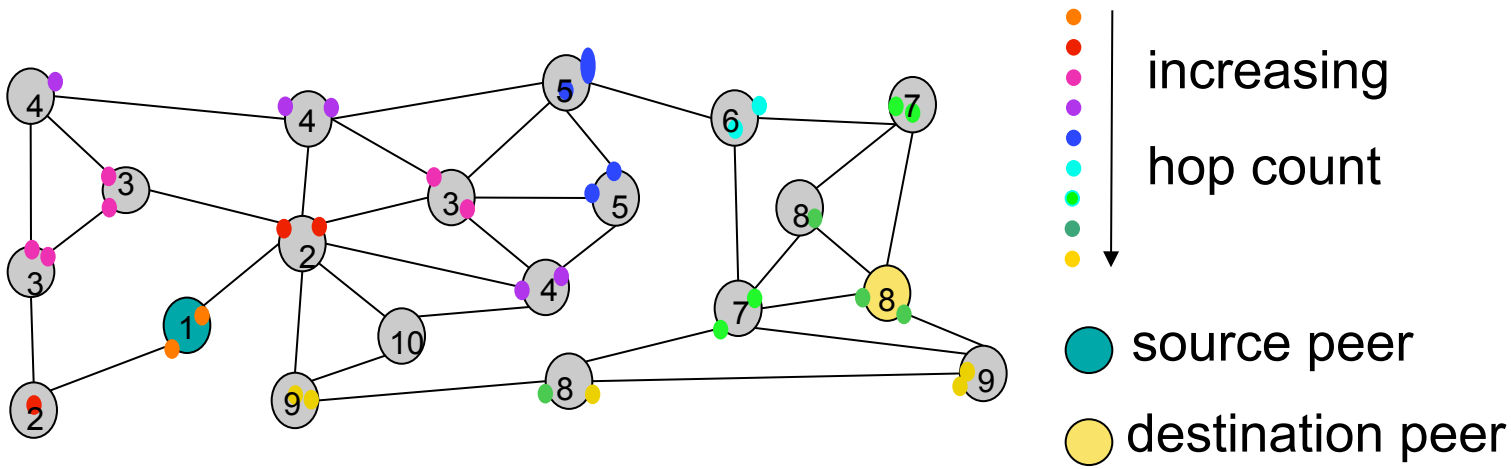
- Forward the query to a randomly selected neighbor
  - Message overhead is reduced significantly
  - Increased latency



- Multiple random walks (k-query messages)
  - Reduces latency
  - Generates more load

- Termination mechanism
  - TTL-based
  - Periodically checking requester before next submission

Example



Random walk with n=2

- (each incoming message is sent twice out)

Overhead

- Smaller, here e.g. 30 messages sent until destination is reached

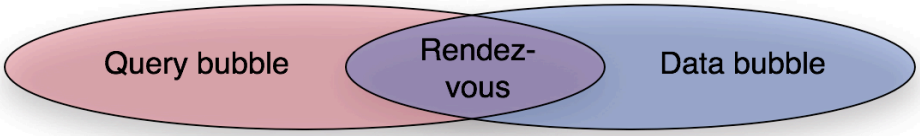
Length of the path found

- e.g.
  - 7 hops

Let‘s have a Rendezvouz

Up to now

- File „uploader“ just keeps the file
- Searching nodes must become active
- Large message overhead



Rendezvouz idea

- Uploader replicates data randomly
- Higher chance to find data, less search messages sufficient

Related question:
How many students do we need in this lecture
in order to have 2 students having the same birthday
with a probability > 50%?

Birthday paradox solution

Number of entries: k

Maximum set size: n (here 365)

Probability of new student having a different birthday than previous students: (n-k)/n

Probability to not match (product of all such cases):

$\Pi_{i=0}^{i=k}$ (n-i)/n

Example:

- Birthday, n=365
- Probability for a match > 0.5 with 23 persons
- 10.000 nodes → 116 nodes needed

Previous persons	P(new day)	P(No equal birthdays)
0	1,000000	1,000000
1	0,997260	0,997260
2	0,994521	0,991796
3	0,991781	0,983644
4	0,989041	0,972864
5	0,986301	0,959538
6	0,983562	0,943764
7	0,980822	0,925665
8	0,978082	0,905376
9	0,975342	0,883052
10	0,972603	0,858859
11	0,969863	0,832975
12	0,967123	0,805590
13	0,964384	0,776897
14	0,961644	0,747099
15	0,958904	0,716396
16	0,956164	0,684992
17	0,953425	0,653089
18	0,950685	0,620881
19	0,947945	0,588562
20	0,945205	0,556312
21	0,942466	0,524305
22	0,939726	0,492703

Search Mechanisms: Rendezvous Point

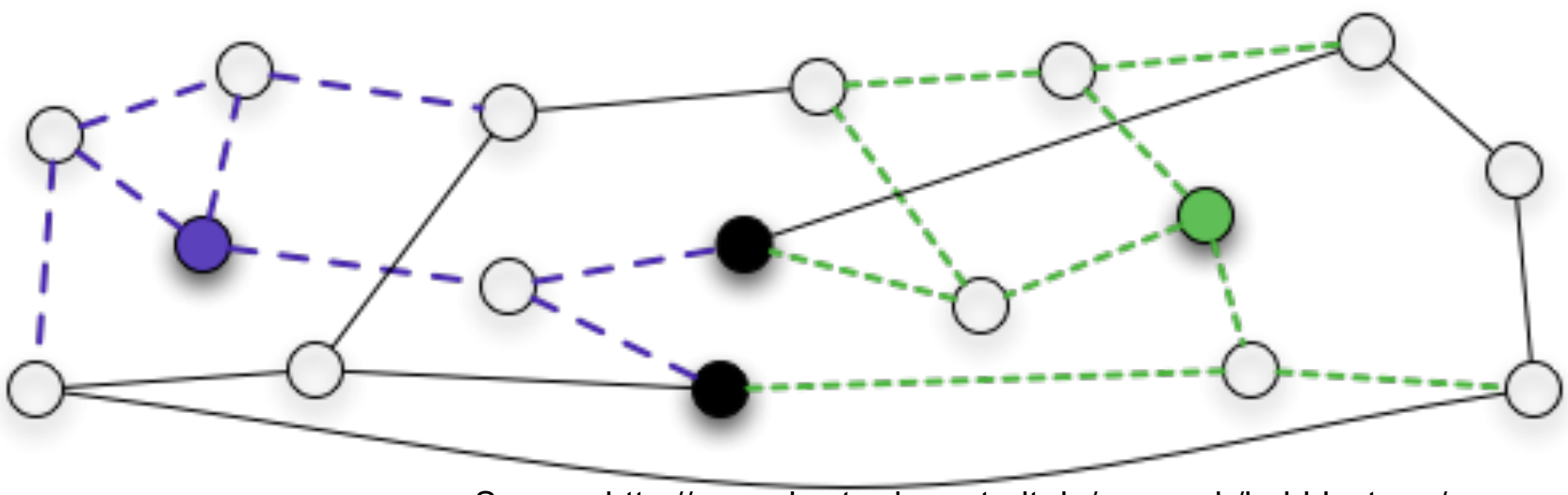
Storing node

- Green/light grey on right side
- Propagate content on all nodes within a predefined range

Requesting node

- Blue/dark grey on left side
- Propagates his query to all neighbors within predefined range

A query hit can be found at the Rendezvous Point (black)



Source: http://www.dvs.tu-darmstadt.de/research/bubblestorm/

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Unstructured P2P Overlay Networks
– Gnutella 0.4

This slide set is based on the lecture "Communication Networks 2" of Prof. Dr.-Ing. Ralf Steinmetz at TU Darmstadt

Homogeneous P2P Overlay: Gnutella - Version 0.4

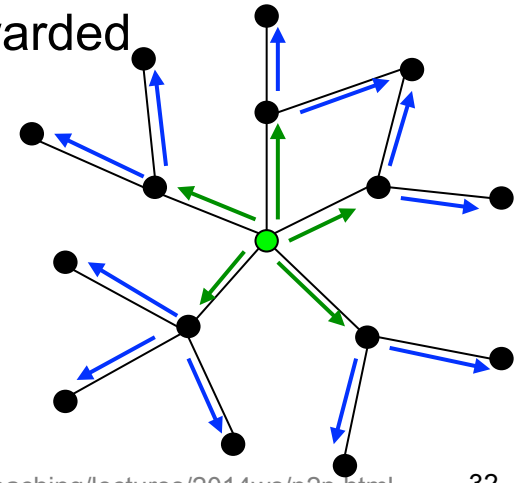
Gnutella Version 0.4

- History: first decentralized p2p overlay
- Developed by Nullsoft (owned by AOL)
- Protocol version 0.4 – homogeneous roles
- Protocol version 0.6 – heterogeneous roles / hierarchical structure

Some Characteristics (of ver. 0.4) :

Message broadcast for node discovery and search requests

- Flooding
  - (to all connected nodes) is used to distribute information
- Nodes recognize message they already have forwarded
  - by their GUID and
  - do not forward them twice



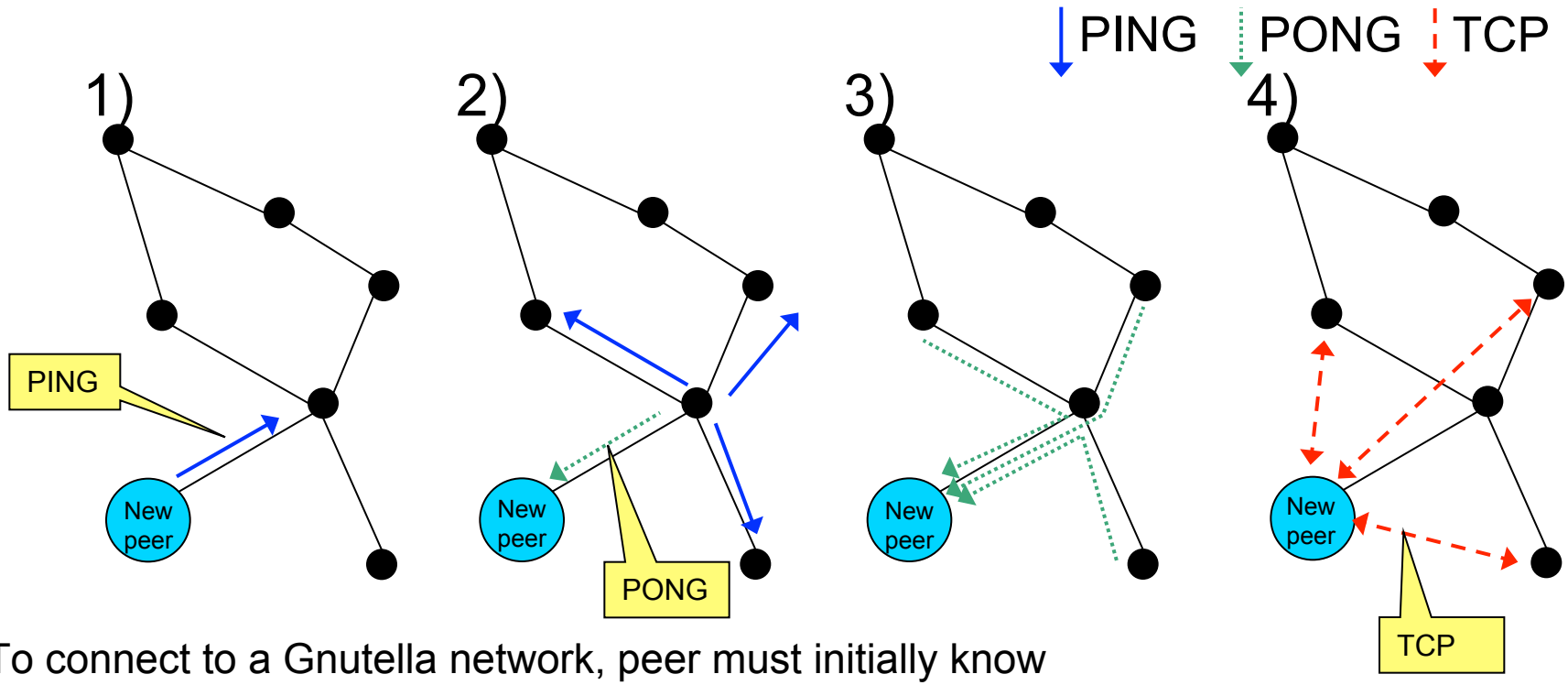
Hop limit by TTL

- Originally TTL = 7

Phases of Protocol 0.4

1. Connecting
   - PING message:
     - Actively discover hosts on the network
   - PONG message:
     - Answer to the PING messages
     - Includes information about one connected Gnutella servent
2. Search
   - QUERY message:
     - Searching the distributed network
   - QUERY HIT message:
     - Response to a QUERY message
     - Can contain several matching files of one servent
3. Data transfer
   - HTTP is used to transfer files (HTTP GET)
     - Not part of the protocol v0.4
   - PUSH message: To circumvent firewalls

Phase 1: Gnutella – Connecting



To connect to a Gnutella network, peer must initially know

- (At least) one member node of the network and connect to it

This/these first member node(s) must be found by other means

- Find first member using other medium (Web, telephone …)
- Nowadays host caches are usually used

Servent connects to a number of nodes

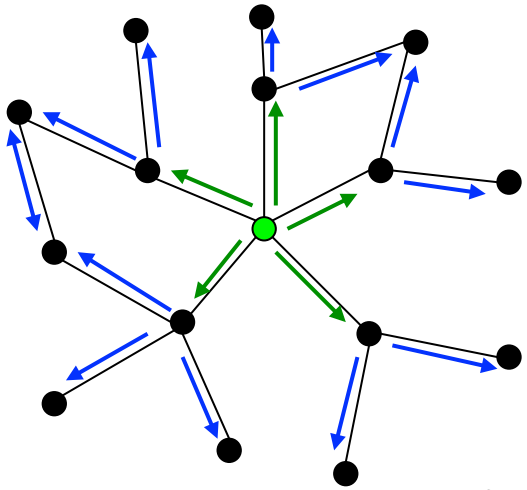
- It got PONG messages from
  - → thus it becomes part of the Gnutella network

Phase 2: Gnutella – Searching

Flooding: QUERY message is distributed to all connected nodes

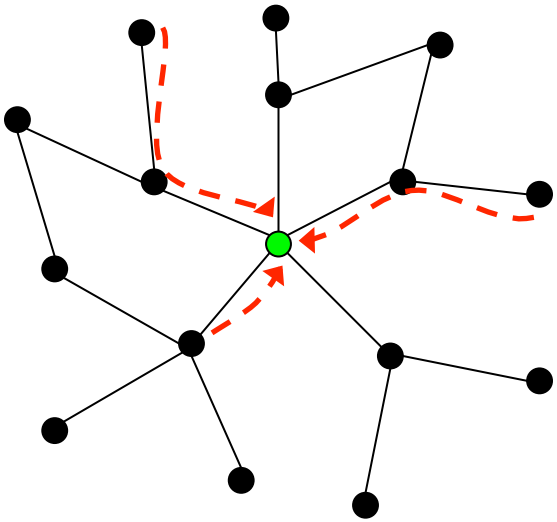
1. A node that receives a QUERY message
   - increases the HOP count field of the message and
   - IF
     - HOP <= TTL (Time To Live)
     - and a QUERY message with this GUID was not received before
   - THEN forwards it to all nodes
     - except the one he received it from

→ Flooding



Phase 2: Gnutella – Searching

2. The node also checks
   - Whether it can answer to this QUERY
   - With a QUERYHIT message
   - If e.g. available files match the search criteria



3. QUERYHIT message
   - Contains
     - The IP address of the sender
     - Information about one or more files that match search criteria
   - Information passed back the same way the QUERY took
     - No flooding
     - No need to establish new links → avoids NAT problem

Phase 3: Gnutella – Data Transfer

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer sets up a HTTP connection

- Actual data transfer is not part of the Gnutella protocol
- HTTP GET is used

Special case:
peer with the file located behind a firewall/NAT gateway

- Downloading peer
  - Cannot initiate a TCP/HTTP connection
  - Can instead send the PUSH message
    - Asking the other peer to initiate a TCP/HTTP connection to it and
    - Then transfer (push) the file via it
  - Does not work if both peers are behind firewalls

Gnutella 0.4: Scalability Issues

A TTL (Time To Live) of 4 hops for the PING messages

- Leads to a known topology of roughly 8000 nodes
- TTL in the original Gnutella client was 7 (not 4)

Gnutella 0.4 suffers from a range of scalability issues, due to

- Fully decentralized approach
- Flooding of messages

Low bandwidth peers easily use up all their bandwidth

- For forwarding PING and QUERY
- No bandwidth for up- and downloads

QUERY	54.8%
PONG	26.9%
PING	14.8%
QUERY HIT	2.8%
PUSH	0.7%

⇨41.7% of the messages just for network discovery

Frequency Distribution of Gnutella Messages (e.g. Portmann et. al.)

→ Breakdown of Gnutella network in August 2000

Gnutella 0.4: How Scalability Issues are tackled

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Mechanisms to overcome the performance problems

- Optimization of connection parameters
  - Number of hops etc
- PING/PONG optimization
  - e.g. C. Rohrs und V. Falco. Limewire: Ping Pong Scheme, April 2001.
    - http://www.limewire.com/index.jsp/pingpong
- Dropping of low-bandwidth connections
  - To move low-bandwidth users to edge of Gnutella Network
- Ultra-Peers
  - similar to KaZaA super nodes
- File hashes to identify files
  - similar to eDonkey

Chosen solution: Hierarchy (Gnutella v. 0.6):

- SuperPeers (like in FastTrack..)
- To cope with load of low bandwidth peers

Gnutella 0.4 Problem: Free Riders

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Free Rider

- Selfish individuals that opt out of a voluntary contribution
  - To the community social welfare
  - I.e. by not sharing any files but downloading from others
  - ... and get away with it
- Exist in big anonymous communities

Study results (since e.g. Adar/Hubermann 2000):

- 70% of the Gnutella users share no files
- 90% answer no queries

Solutions

- Increase incentives for sharing
  - peers only accept connections / forward messages from peers that share content
  - but: e.g., how to verify? Quality of the content?
- Micro-payment

Exercise 1

Peer-to-Peer Systems
Exercise 1
Published: 23.10.2014
Presentation date: 30.10.2014

Peer-to-Peer Systems – Exercise
Winter Term 2014/2015

General Remarks

Welcome to the exercise for the lecture Peer-to-Peer Systems.
Please follow the general remarks regarding the organization of the exercise.

- The lecture's website is to be found here: `http://tsn.hhu.de/teaching/lectures/2014ws/p2p.html`
- For further inquiries, please contact the lecturer under the following email address: `graffi@cs.uni-duesseldorf.de`

Problem 1.1 - Gnutella Messages

Please consider the Gnutella topology as depicted in the Figure 1. All peers have a local view on the topology which they maintain. The peers further maintain a list of messages they forwarded recently in order to avoid unnecessary retransmissions. In this exercise we assume a round-based communication, thus messages are received in parallel, sent in parallel and received in the next round.

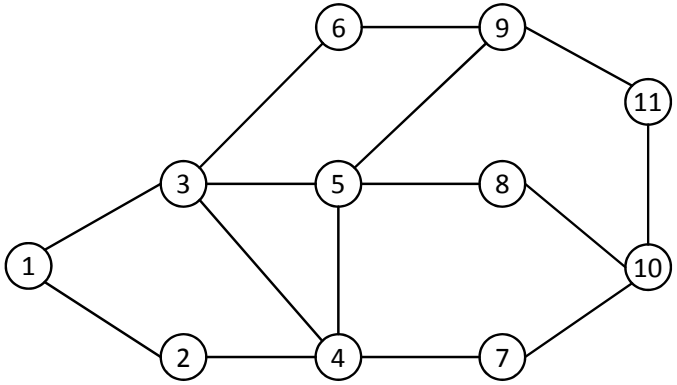


Abbildung 1: Gnutella Topology

a) Query Forwarding in Gnutella

Peer 1 attempts to search for the keyword „P2P“ using a TTL of 2. The Hop count is increased by one each time the query is received. The queries are propagated in the network until the Hop > TTL runs out. List all messages transferred in the network according to the message template in following table (Table 1). The hop count denotes the value at the receiver side.

Round	Sender	Receiver	Hop
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

Tabelle 1: Message Format

b) Routing of QUERYHIT Messages in Gnutella

Now assume that the nodes 2, 3, 8 and 11 have files matching the keyword „P2P“. List the QUERYHIT messages that occur after the query in a) has been stated. Use the same message type (except the Hop) and use also the same rounds.

c) No State Information in Gnutella

Assume that in order to save storage on peers, no state information for query forwarding is hold. Describe at one example in Figure 1 the problem that arises.

Exercise 1

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Problem 1.2 - Scalability of Gnutella

In the following task we will have a closer look on the scalability of Gnutella. First we are going to calculate the number of reachable nodes for a single request. Afterwards, we will have a look on the total traffic caused by a single query and its response. Please use the notation as shown in the table below in the following task.

Symbol	Description
n	Number of connections each client has to its neighbors
t	Time-to-live counter
s	Size in bytes of single request
a	Mean percentage of users who typically share content
b	Mean percentage of users who typically have responses to search queries
l	Mean amount of data of a query response
w	total bandwidth of the network in MBps
$R(a, b, l)$	A function representing the Response Factor, a constant value that describes the product of the percentage of users responding and the amount of data generated by each user.
$f(n, x, y)$	A function describing the maximum number of reachable users that are at least x hops away, but no more than y hops away.
$h(n, t, s)$	A function describing the maximum amount of traffic generated by relaying a transmission of s bytes given any n and t. Generation is defined as the formulation and outbound delivery of data
$k(n, t, R)$	A function describing the maximum amount of traffic generated in response to a search query, including relayed data, given any n and t and Response Factor R.

a) Formula for Number of Users

Given a node A with n outgoing links. Please derive a formula $f(n, x, y)$ for the maximum number of reachable users that are at least x hops away, but no more than y hops away from node A. Assume that the number of links each user maintains to its neighbors is n and that we do not discover duplicate nodes on each of the n outgoing links in case of a query request.

b) Calculation of Number of Users

Please calculate the number of reachable users for $t = 7$ and $n = 4$ as well as for $t = 7$ and $n = 8$.

c) Formula for Query Traffic for a Single Query Request

Please derive a formula $h(n, t, s)$ for the total traffic generated by a single query request assuming a query size of s, a TTL counter of t, and an out-degree of n. Please take into account all outbound and inbound transmissions. Hint: A request sent from a Node A to a Node B causes outbound traffic at Node A as well as inbound traffic at Node B.

d) Query Traffic for a Single Query Request

Please calculate the total amount of traffic caused for a singe query request given the size of $s = 100$ bytes given $t = 7$ and $n = 4$ as well as for $t = 7$ and $n = 8$.

e) Total Traffic for a Single Query-Response-Cycle

Given the mean percentage of users who typically share content $a = 0.5$, mean percentage of users who typically have responses to search queries $b = 0.3$, and the mean size of a query response $l = 300$ bytes, the response factor R can be calculated as follows:

$$R(a, b, l) = a * b * l \tag{1}$$

Then the total amount of traffic generated by the answering nodes is given as:

$$k(n, t, R(a, b, l)) = R(a, b, l) * \sum_{i=1}^{t} f(n, i, i) * i \tag{2}$$

Please calculate the total amount of traffic caused for a complete query-response-cyc given the size of $s = 100$ bytes given $t = 7$ and $n = 4$ as well as for $t = 7$ and $n = 8$.

f) Total Bandwidth Consumption

Finally, let us assume that we have 10 query requests per second in a Gnutella network. How much total bandwidth w is required in the whole network to solve all the queries assuming the same values for t, n, s, and R as in Task E?

g) Conclusions

Which conclusions can you draw from the results above? How can the scalability of the network be improved?

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer-to-Peer Systems

Unstructured P2P Overlay Networks
– Rendezvouz-based Search

This slide set is based on the lecture "Communication Networks 2" of Prof. Dr.-Ing. Ralf Steinmetz at TU Darmstadt

Rendezvous-based P2P Overlays

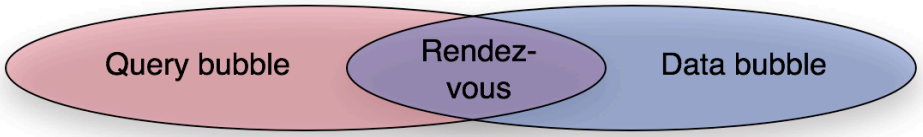
Rendezvous idea

- Content is announced in a region (bubble) of the P2P network
- Queries flood just a region (bubble) of the P2P network
- Announcements and queries meet at a rendezvous point

Replicate both queries and data

- O($\sqrt{n}$) copies each (hidden constants unequal)

Data and queries rendezvous in the network



Source: Terpstra, Leng, Lehn – Short Presentation Bubblestorm

Rendezvous-based P2P Overlay: Bubblestorm

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Random Topology

- Peer neighbors are chosen randomly
- Allows efficient sampling of peers at random

Topology Measurement

- To calculate bubble sizes the network size must be known
- Computes network size and statistics through gossiping

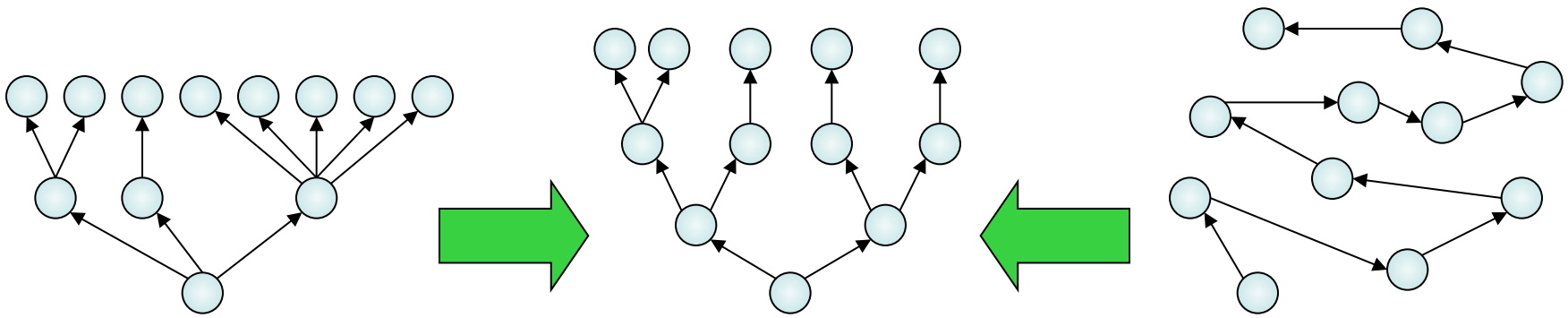
Bubblecast

- Replicates queries/data onto peers quickly
- Intelligent flooding to / in the bubble

Bubble Maintainer

- Preserves the correct number of replicas

Bubblecast Motivation



Flooding

+ low latency
+ reliable
- imprecise node count
- unbalanced link load

Bubblecast

+ low latency
+ reliable
+ precise node count
+ balanced link load

Random Walk

- high latency
- unreliable
+ precise length
+ balanced link load

node counter (not hops)
fixed branch factor

branch in every step

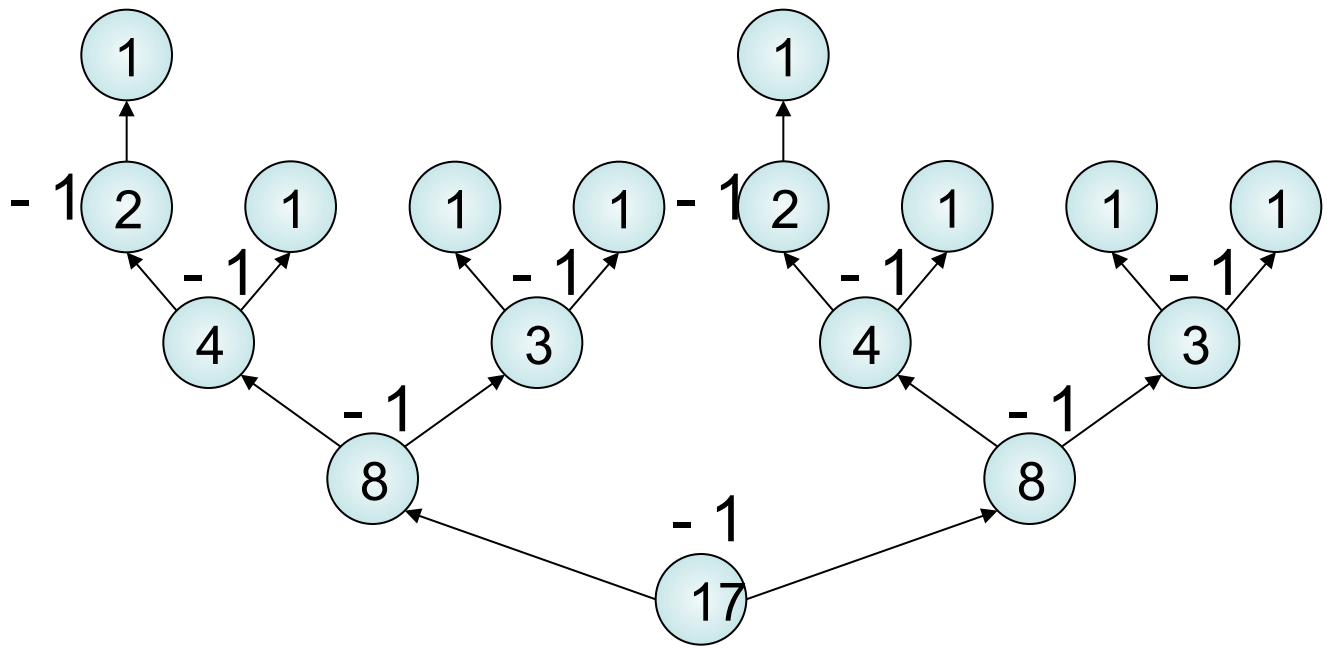
Source: Terpstra, Leng, Lehn – Short Presentation Bubblestorm

Example Bubblecast Execution

Bubblecast: Announcement / query in the bubble

Procedure:

- A counter specifies the number of replicas to create
- Decrement the counter for matching locally
- Split the counter between two neighbors
- Counters are always integral
- Final routing depth differs by at most one hop



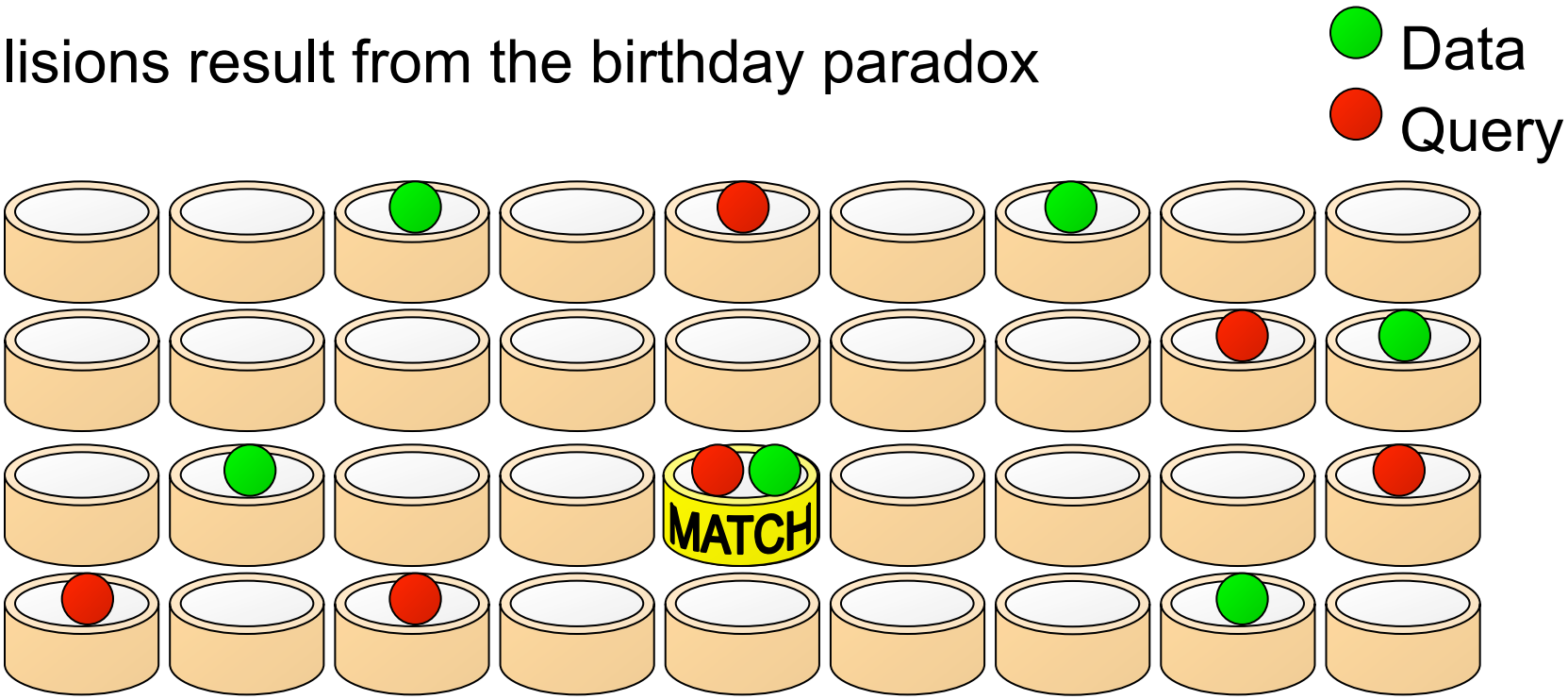
BubbleStorm: Random Replication

Place data replicas on random nodes

Nodes evaluate query replicas on all stored data

- Where both data and query go, matches are found

Collisions result from the birthday paradox

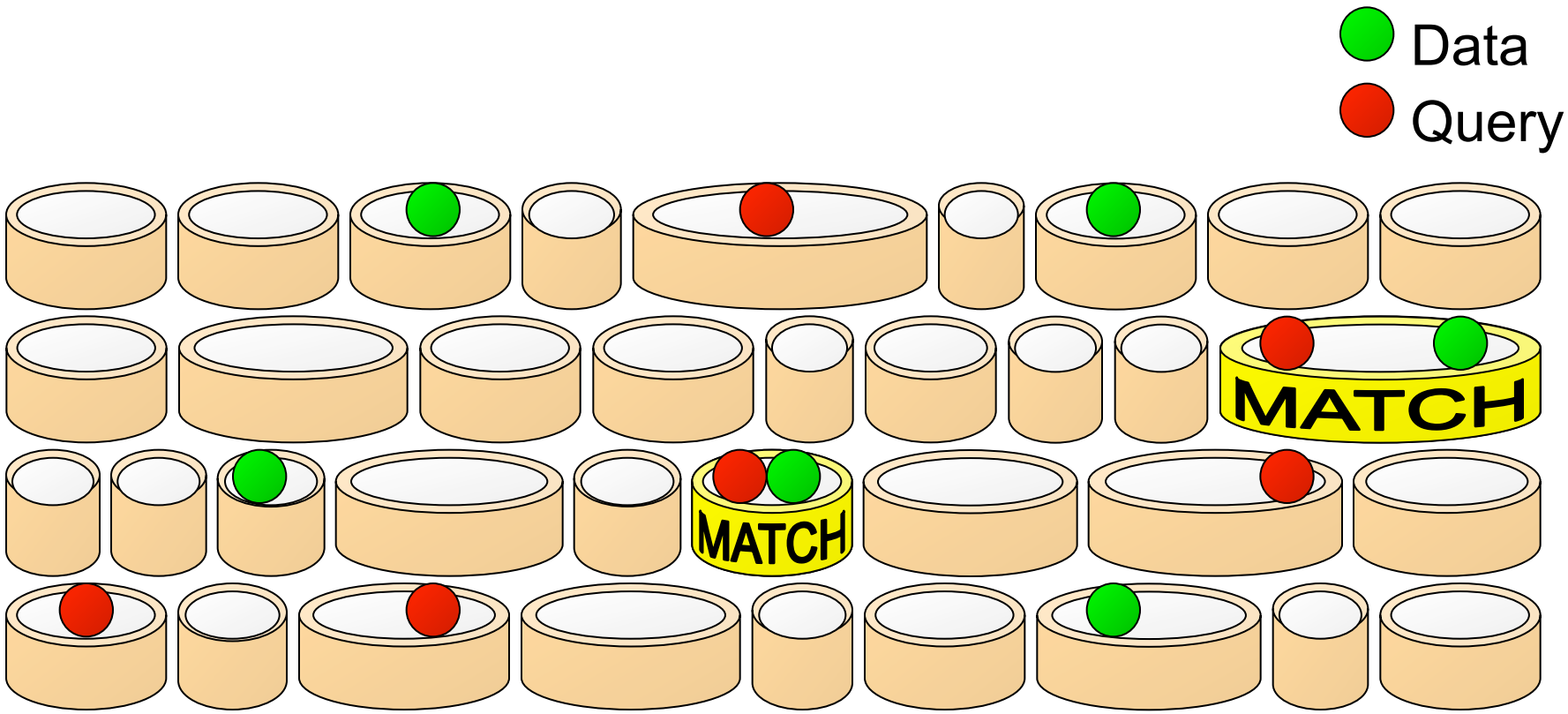


Source: Terpstra, Leng, Lehn – Short Presentation Bubblestorm

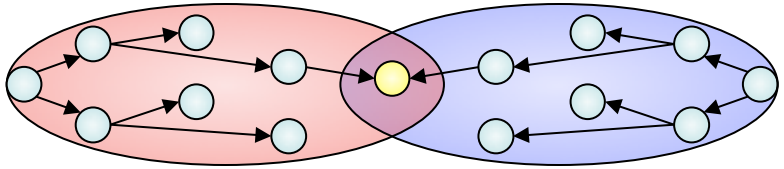
BubbleStorm: Exploiting Heterogeneity

Node degree is chosen proportional to bandwidth

- As random walks and bubblecasts follow edges with equal probability
- Utilization will be balanced for heterogeneity



Bubblecast Properties



Used for query and data replication

Fixed branch factor balances load

- Same stationary distribution as a random walk

Counter for edges crossed, not hops

- Precisely controls replica count

Logarithmic routing depth

- Slightly deeper than flooding

Message loss reduces replication by log(size)

Samples random nodes

- … due to random topology

Complexity and Correctness

BubbleStorm costs roughly $c\sqrt{n}$ bandwidth / operation to provide exhaustive search with $\mathrm{P(failure)} < e^{-c^2}$

c	1	2	3	4
P(success)	63.21%	98.17%	99.99%	99.99999%

The full equation ($e^{-c^2+c^3 HY}$) is complicated by

- Heterogeneous peer capacity (H)
- Dependent sampling (due to repeated withdrawals)
- Unequal query and post traffic (Y; BS optimizes this)
- Full details in the paper