Topics in Database Systems: Data Management in Peer-to-Peer Systems

Unstructured Peer-to-Peer Systems

Γιατί θα μιλήσουμε σήμερα ..

1. ΜΕΡΟΣ 1: Γενική Εισαγωγή σε Αδόμητα (Unstructured) Συστήματα Ομότιμων Κόμβων (ΣΟΚ) και κάποια γενικά για ΣΟΚ
2. ΜΕΡΟΣ 2: Ένα Παράδειγμα Χρήσης Ευρετηρίων σε Αδόμητα ΣΟΚ – Routing Indexes

Ασκήσεις για 29/3

1. Θα διορθώσετε το προηγούμενο σύνολο μέχρι τη Δευτέρα (28/3)
2. Θα απαντήσετε σε 2-3 ερωτήσεις πάνω στη σημερινή ύλη (στο Μέρος 2) (θα ανακοινωθούν αύριο)

Περιμένω τις διαφάνειες κάποιων από 8/3

Topics in Database Systems: Data Management in Peer-to-Peer Systems

Unstructured Peer-to-Peer Systems

PART I (assorted)

Unstructured Peer-to-Peer Systems

Based on

"Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems"

Karl Aberer & Manfred Hauswirth

ICDE02 Tutorial

"Architectures and Algorithms for Internet-Scale (P2P) Data Management"

Joe Hellerstein

VLDB 2004 Tutorial

"Open Problems in Data-Sharing Peer-to-Peer Systems",

Neil Daswani, Hector Garcia-Molina and Beverly Yang. In ICDT, 2003.

Θα βάλω αντίγραφα στη σελίδα

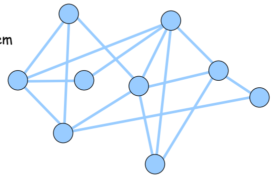
What is a P2P System?

- Multiple sites (at edge)
- Distributed resources
- Sites are autonomous (different owners)
- Sites are both clients and servers
- Sites have equal functionality

P2P Purity

What is P2P?

- Every participating node acts as both a client and a server ("servent")
- Every node "pays" its participation by providing access to (some of) its resources
- Properties:
  - no central coordination
  - no central database
  - no peer has a global view of the system
  - global behavior emerges from local interactions
  - all existing data and services are accessible from any peer
  - peers are autonomous
  - peers and connections are unreliable



Overlay Networks

- P2P applications need to:
  - Track identities & (IP) addresses of peers
    - May be many!
    - May have significant Churn (update rate)
    - Best not to have n^2 ID references
  - Route messages among peers
    - If you don't keep track of all peers, this is *"multi-hop"*

- This is an *overlay network*
  - Peers are doing both naming and routing
  - IP becomes "just" the low-level transport
    - *All the IP routing is opaque*

P2P Cooperation Models

- Centralized model
  - global index held by a central authority (single point of failure)
  - direct contact between requestors and providers
  - Example: Napster
- Decentralized model
  - Examples: Freenet, Gnutella
  - no global index, no central coordination, global behavior emerges from local interactions, etc.
  - direct contact between requestors and providers (Gnutella) or mediated by a chain of intermediaries (Freenet)
- Hierarchical model
  - introduction of "super-peers"
  - mix of centralized and decentralized model
  - Example: DNS

Many New Challenges

- Relative to other parallel/distributed systems
  - Partial failure
  - Churn
  - Few guarantees on transport, storage, etc.
  - Huge optimization space
  - Network bottlenecks & other resource constraints
  - No administrative organizations
  - Trust issues: security, privacy, incentives
- Relative to IP networking
  - Much higher function, more flexible
  - Much less controllable/predictable

Why Bother? Not the Gold Standard

- Given an infinite budget, would you go p2p?

- Highest performance? No.
  - Hard to beat hosted/managed services
  - p2p Google appears to be infeasible [Li, et al. IPTPS 03]

- Most Resilient? Hmmmm.
  - In principle more resistant to DoS attacks, etc.
  - Take, Chord: A node entering multiple times in the ring with different identities, control much f the traffic
  - Today, still hard to beat hosted/managed services
    - Geographically replicated, hugely provisioned
    - People who "do it for dollars" today don't do it p2p

Why Bother II: Positive Lessons from Filestealing

- P2P enables *organic scaling*
  - Vs. the top few killer services -- no VCs required!
  - Can afford to "place more bets", try wacky ideas
- Centralized services engender scrutiny
  - Tracking users is trivial
  - Provider is liable (for misuse, for downtime, for local laws, etc.)
- Centralized means business
  - Need to pay off startup & maintenance expenses
  - Need to protect against liability
  - Business requirements drive to particular short-term goals
    - *Tragedy of the commons*

Why Bother III? Intellectual motivation

- Heady mix of theory and systems
  - Great community of researchers have gathered
  - Algorithms, Networking, Distributed Systems, Databases
  - Healthy set of publication venues
    - IPTPS workshop, P2P conference

    (classical venues (DB: VLDB, SIGMOD, ICDE DC: ICDCS, etc)

Infecting the Network, Peer-to-Peer

- The Internet is hard to change.
- But Overlay Nets are easy!
  - P2P is a wonderful "host" for infecting network designs
  - The "next" Internet is likely to be very different
    - "Naming" is a key design issue today
    - Querying and data independence key tomorrow?
- Don't forget:
  - The Internet was originally an overlay on the telephone network
  - There is no money to be made in the bit-shipping business

- A modest goal for DB research:
  - Don't query the Internet.

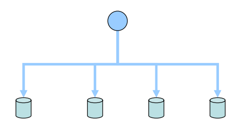
Infecting the Network, Peer-to-Peer

A modest goal for DB research:
- Don't query the Internet.

Be the Internet.

Distributed Databases

- Fragmenting large databases (e.g., relational) over physically distributed nodes
- Efficient processing of complex queries (e.g., SQL) by decomposing them
- Efficient update strategies (e.g., lazy vs. eager)
- Consistent transactions (e.g., 2 phase commit)
- Normally approaches rely on central coordination



Distributed Databases vs. Peer-to-Peer

- Data distribution is a key issue for P2P systems
- Distribution Transparency
- Data Allocation and Fragmentation
- Advanced (SQL?) Query Processing
- Transactions

Main P2P Design Requirements

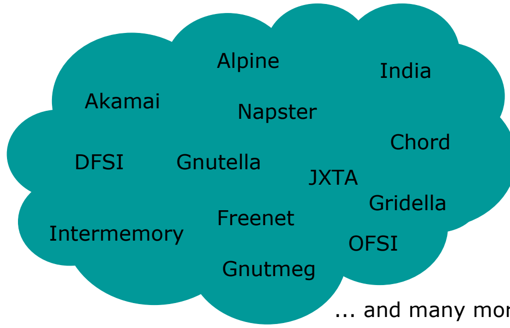
- Resource discovery
- Managing updates
- Scalability
- Robustness and fault tolerance
- Trust assessment and management

Usage Patterns to position P2P

Discovering information is the predominant problem

- Occasional discovery: search engines
  - ad hoc requests, irregular
  - E.g., new town — where is the next car rental?
- Notification: event-based systems
  - notification for (correlated) events (event patterns)
  - E.g., notify me when my stocks drop below a threshold
- Regular discovery: P2P systems
  - find certain type of information on a regular basis
  - E.g., search for MP3 files of Jethro Tull regularly
- Continuous information feed: push systems
  - subscription to a certain information type
  - E.g., sports channel, updates are sent as soon as available

The P2P Cloud



... and many more ...

What is P2P?

pastry can jxta fiorana
napster united devices freenet
aim ocean store open cola
gnutella farsite ? netmeeting
maorpheus icq ebay
uddi grove bearshare limewire seti@home
kazaa jabber popular power
folding@home tapestry
process tree chord mojo nation

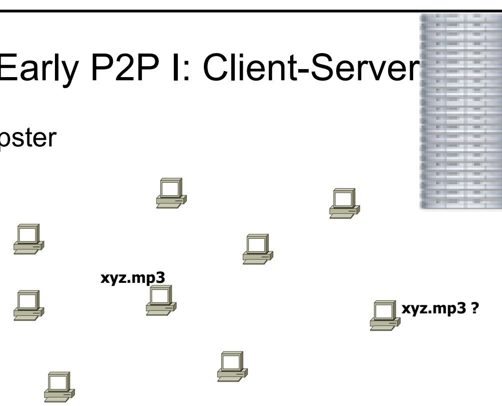
Early P2P

Unstructured P2P Systems

- Napster
- Gnutella
- Freenet

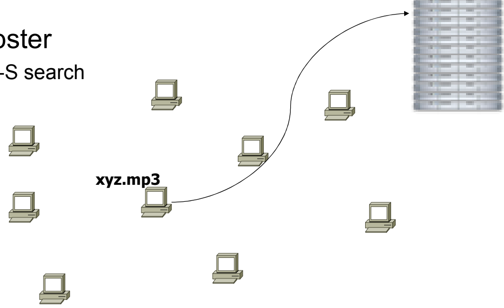
Early P2P I: Client-Server

- Napster



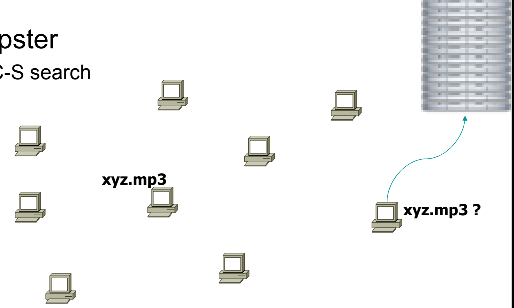
Early P2P I: Client-Server

- Napster
  - C-S search



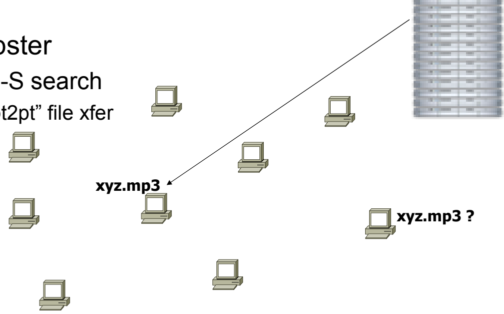
Early P2P I: Client-Server

- Napster
  - C-S search



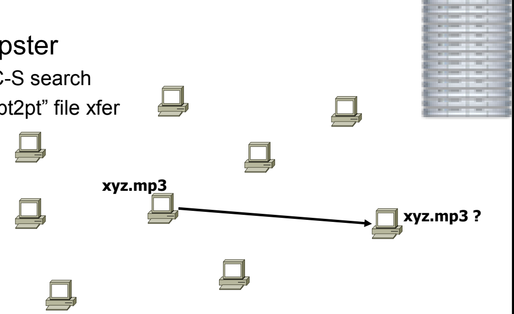
Early P2P I: Client-Server

- Napster
  - C-S search
  - “pt2pt” file xfer



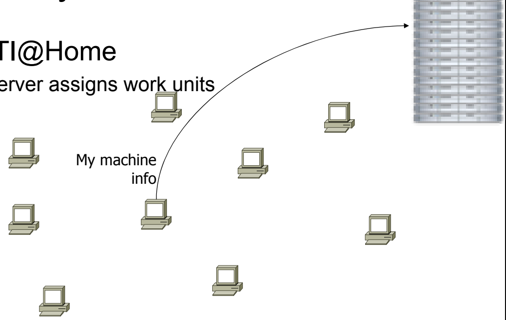
Early P2P I: Client-Server

- Napster
  - C-S search
  - “pt2pt” file xfer



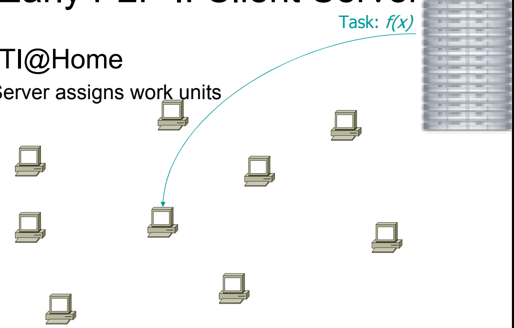
Early P2P I: Client Server

- SETI@Home
  - Server assigns work units



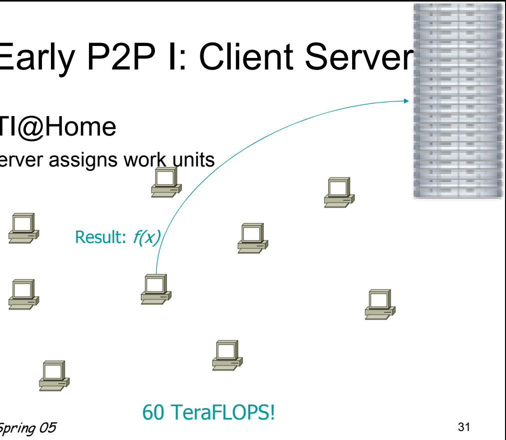
Early P2P I: Client Server

- SETI@Home
  - Server assigns work units



Early P2P I: Client Server

- SETI@Home
  - Server assigns work units



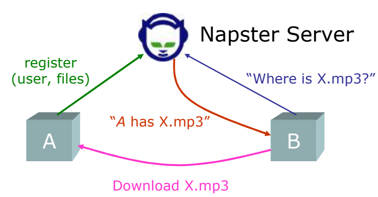
More on Napster: A brief History

- **May 1999:** Napster Inc. file share service founded by Shawn Fanning and Sean Parker
- **Dec 7 1999:** Recording Industry Association of America (RIAA) sues Napster for copyright infringement
- **April 13, 2000:** Heavy metal rock group Metallica sues Napster for copyright infringement
- **April 27, 2000:** Rapper Dr. Dre sues Napster
- **May 3, 2000:** Metallica's attorney claims 335,000 Internet users illegally share Metallica's songs via Napster
- **July 26, 2000:** Court orders Napster to shut down
- **Oct 31, 2000:** Bertelsmann becomes a partner and drops lawsuit
- **Feb 12, 2001:** Court orders Napster to cease trading copyrighted songs and to prevent subscribers to gain access to content on its search index that could potentially infringe copyrights
- **Feb 20, 2001:** Napster offers $1 billion to record companies (rejected)
- **March 2, 2001:** Napster installs software to satisfy the order

Napster: System Architecture

- Central (virtual) database which holds an index of offered MP3/WMA files
- Clients(!) connect to this server, identify themselves (account) and send a list of MP3/WMA files they are sharing (C/S)
- Other clients can search the index and learn from which clients they can retrieve the file (P2P)
- Combination of client/server and P2P approaches
- First time users must register an account

Napster: Communication Model



Napster: The Protocol [Drscholl01]

- The protocol was never published openly and is rather complex and inconsistent
- OpenNap have reverse engineered the protocol and published their findings
- TCP is used for C/S communication
- Messages to/from the server have the following format:

length	type	data

Byte offset 0 1 2 3 4 ..... n

- length specifies the length of the data portion
- type defines the message type
- data: the transferred data
  - plain ASCII, in many cases enclosed in double quotes (e.g., filenames such as "song.mp3" or client ids such as "nap v0.8"

Sample Messages - 1

Type	C/S	Description	Format
0	S	Error message	<message>
2	C	Login	<nick><pwd><port><client info><link type>
3	S	Login ack	<user's email>
5	S	Auto-upgrade	<new version><http-hostname:filename>
6	C	New user login	<nick><pwd><port><client info><speed> <email address>
100	C	Client notification of shared file	"<filename>"<md5><size><bitrate> <frequency><time>
200	C	Search request	[FILENAME CONTAINS "artist name"] MAX_RESULTS <max> [FILENAME CONTAINS <song] [LINESPEED <comp> <link type>] [BITRATE <comp> "bit rate"] [FREQ <comp> "freq"] [WMA-FILE] [LOCAL_ONLY]
201	S	Search response	"<filename>"<md5><size><bit rate> <frequency><length><nick><ip address>
202	S	End of search response	(empty)

Sample Messages - 2

Type	C/S	Description	Format
203	C	Download request	<nick> "<filename>"
204	S	Download ack	<nick><ip><port> "<filename>" <md5> <linespeed>
206	S	Peer to download not available	<nick> "<filename>"
209	S	Hotlist user signed on	<user><speed>
211	C	Browse a user's files	<nick>
212	S	Browse response	<nick> "<filename>"<md5><size> <bit rate><frequency><time>
213	S	End of browse list	<nick>[<ip address>]
500	C	Push file to me (firewall problem)	<nick> "<filename>"
501	S	Push ack (to other client)	<nick><ip address><port> "<filename>" <md5><speed>

Client-Client Communication - 1

- Normal download (*A* downloads from *B*):
  - *A* connects to *B*'s IP address/port as specified in the 204 message returned by the server (response to 203)
  - *B* sends the ASCII character "1"
  - *A* sends the string "GET"
  - *A* sends <mynick> "<filename>" <offset>
  - *B* returns the file size (not terminated by any special character!) or an error message such as "FILE NOT SHARED"
  - *A* notifies the server that the download is ongoing via a 218 message; likewise *B* informs the server with a 220 message
  - Upon successful completion *A* notifies the server with a 219 message; likewise *B* informs the server with a 221 message

Client-Client Communication - 2

- Firewalled download (*A* wants to download from *B* who is behind a firewall):
  - *A* sends a 500 message to the server which in turn sends a 501 message (holding *A*'s IP address and data port) to *B*
  - *B* connects *A* according to the 501 message
  - *A* sends the ASCII character "1"
  - *B* sends the string "SEND"
  - *B* sends <mynick> "<filename>" <size>
  - *A* returns the byte offset at which the transfer should start (plain ASCII characters) or an error message such as "INVALID REQUEST"
  - *A* notifies the server that the download is ongoing via a 218 message; likewise *B* informs the server with a 220 message
  - Upon successful completion *A* notifies the server with a 219 message; likewise *B* informs the server with a 221 message

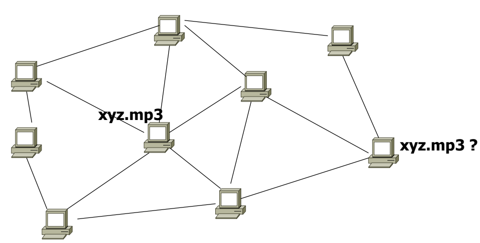
Napster: Further Services

- Additionally to its search/transfer features the Napster client offers:
  - A chat program that allows users to chat with each others in forums based on music genre, etc.
  - A audio player to play MP3 files from inside Napster
  - A tracking program to support users in keeping track of their favorite MP3s for later browsing
  - Instant messaging service
- Most of the message types in the protocol deal with hotlist, chat room, and instant messages

Napster: Summary

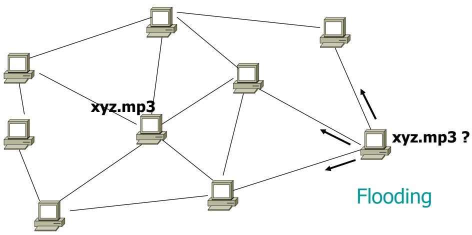
- (Virtually) centralized system
  - single point of failure $\Rightarrow$ limited fault tolerance
  - limited scalability (server farms with load balancing)
- Protocol is complicated and inconsistent
- Querying is fast and upper bound for the duration can be given
- "Topology is known"
- Reputation of peers is not addressed
- Many add-on services users like

Early P2P II: Flooding on Overlays

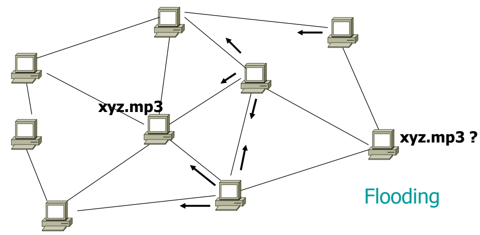


An overlay network. "Unstructured".

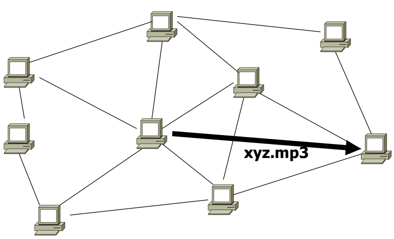
Early P2P II: Flooding on Overlays



Early P2P II: Flooding on Overlays

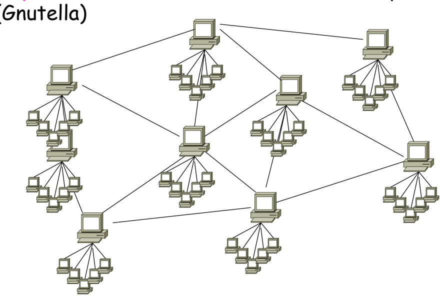


Early P2P II: Flooding on Overlays



Early P2P II.v: "Ultrapeers"

Ultrapeers can be installed (KaZaA) or self-promoted (Gnutella)



Gnutella: A brief History

- Developed in a 14 days "quick hack" by Nullsoft (winamp)
- Originally intended for exchange of recipes
- Timeline:
  - Published under GNU General Public License on the Nullsoft web server
  - Taken off after a couple of hours by AOL (owner of Nullsoft)
  - This was enough to "infect" the Internet
  - Gnutella protocol was reverse engineered from downloaded versions of the original Gnutella software
  - Third-party clients were published and Gnutella started to spread

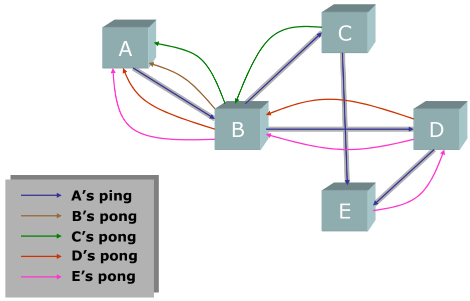
Gnutella: System Architecture

- No central server
  - cannot be sued (Napster)
- Constrained broadcast
  - Every peer sends packets it receives to all of its peers (typically 4)
  - Life-time of packets limited by time-to-live (TTL) (typically set to 7)
  - Packets have unique ids to detect *loops*
- Hooking up to the Gnutella systems requires that a new peer knows at least one Gnutella host
  - gnutellahosts.com:6346
  - Outside the Gnutella protocol specification

Gnutella: Protocol Message Types

Type	Description	Contained Information
Ping	Announce availability and probe for other servents	None
Pong	Response to a ping	IP address and port# of responding servent; number and total kb of files shared
Query	Search request	Minimum network bandwidth of responding servent; search criteria
QueryHit	Returned by servents that have the requested file	IP address, port# and network bandwidth of responding servent; number of results and result set
Push	File download requests for servents behind a firewall	Servent identifier; index of requested file; IP address and port to send file to

Gnutella: Meeting Peers (Ping/Pong)



The Protocol behind: Descriptors

- Meeting
  - GNUTELLA CONNECT/0.4\n\n
  - GNUTELLA OK\n\n
- "Descriptor header" (general packet header)

Byte offset	Descriptor ID	Payload Descriptor	TTL	Hops	Payload Length
	0 15	16	17	18	19 22

- Descriptor ID: 16 byte unique id
- Payload descriptor: packet type (e.g., 0x00 = Ping)
- TTL: the number of times the descriptor will be forwarded
- Hops: TTL(0) = TTL(i) + Hops(i)
- Payload length: the length of the descriptor immediately following this header

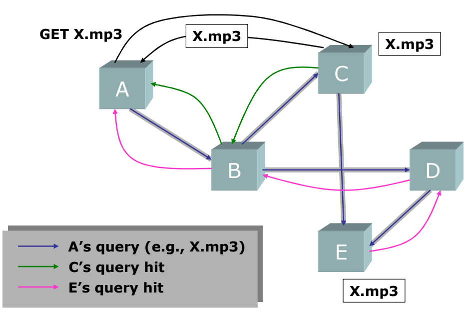
Ping/Pong Descriptors

- Ping (0x00): Descriptor header with payload 0x00
- Pong (0x01):

Byte offset	Port	IP address	Number of files shared	Number of kilobytes shared
	0 1	2 5	6 9	10 13

- Port: on which the responding host can accept connections
- IP address: of the responding host
- Number of files shared
- Number of kilobytes shared

Gnutella: Searching (Query/QueryHit/GET)



Query Descriptor

- Query (0x80):

Byte offset	Minimum speed	Search criteria
	0 1	2 ....

- Minimum speed: the minimum network bandwidth of the servent (in kb/s) that should respond to this query
- Search criteria: a null (i.e., 0x00) terminated string; the maximum length of this string is bounded by the "Payload length" field of the descriptor header.

QueryHit Descriptor (0x81)

Number of hits	Port	IP address	Speed	Result set	Servent identifier

Byte offset 0 1 2 3 6 7 10 11 …. n n+16

– Number of hits: in the result set
– Port: on which the responding host can accept connections
– IP address: of the responding host
– Speed: of the responding host (in kb/s)
– Servent identifier: 16-byte string uniquely identifying the servent
– Result set (*number of hits* records)

File index	File size	File name

Byte offset 0 3 4 7 8 ….

- File index: a number assigned by the responding host to uniquely identify the file matching the corresponding query
- File size: size of the file (in bytes)
- File name: double null (0x0000) terminated name of the file

File Downloads

- Out of band via simplified HTTP
- Connect to IP/address given in QueryHit
- Example:

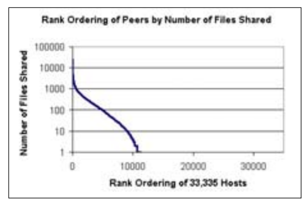
```
GET /get/2468/Foobar.mp3/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0\r\n
User-Agent: Gnutella\r\n
\r\n
```

```
HTTP 200 OK\r\n
Server: Gnutella\r\n
Content-type: application/binary\r\n
Content-length: 4356789\r\n
\r\n
<data> ...
```


Free-riding on Gnutella [Adar00]

- 24 hour sampling period:
  - 70% of Gnutella users share no files
  - 50% of all responses are returned by top 1% of sharing hosts
- A social problem not a technical one
- Problems:
  - Degradation of system performance: collapse?
  - Increase of system vulnerability
  - "Centralized" ("backbone") Gnutella ⇔ copyright issues?
- Verified hypotheses:
  - H1: A significant portion of Gnutella peers are free riders.
  - H2: Free riders are distributed evenly across domains
  - H3: Often hosts share files nobody is interested in (are not downloaded)

Free-riding Statistics - 1 [Adar00]



H1: Most Gnutella users are free riders

Of 33,335 hosts:

– 22,084 (66%) of the peers share no files
– 24,347 (73%) share ten or less files
– Top 1 percent (333) hosts share 37% (1,142,645) of total files shared
– Top 5 percent (1,667) hosts share 70% (1,142,645) of total files shared
– Top 10 percent (3,334) hosts share 87% (2,692,082) of total files shared

Free-riding Statistics - 2 [Adar00]

H3: Many servents share files nobody downloads

Of 11,585 sharing hosts:

- Top 1% of sites provide nearly 47% of all answers
- Top 25% of sites provide 98% of all answers
- 7,349 (63%) never provide a query response

Free Riders

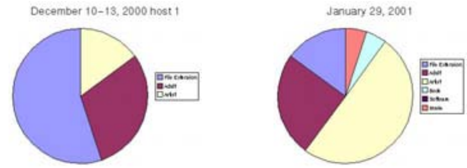
- Filesharing studies
  - Lots of people download
  - Few people serve files
- Is this bad?
  - If there's no incentive to serve, why do people do so?
  - What if there are strong disincentives to being a major server?

Simple Solution: Thresholds

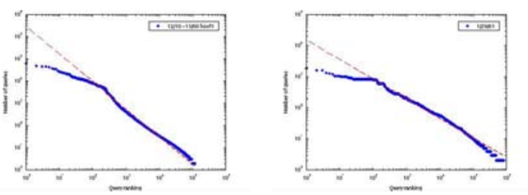
- Many programs allow a threshold to be set
  - Don't upload a file to a peer unless it shares > *k* files
- Problems:
  - What's *k*?
  - How to ensure the shared files are interesting?

Categories of Queries [Sripanidkulchai01]

Categorized top 20 queries



Popularity of Queries [Sripanidkulchai01]



- Very popular documents are approximately equally popular
- Less popular documents follow a Zipf-like distribution (i.e., the probability of seeing a query for the i^{th} most popular query is proportional to **$1/(i^{alpha})$**)
- Access frequency of web documents also follows Zipf-like distributions ⇒ caching might also work for Gnutella

Topology of Gnutella [Jovanovic01]

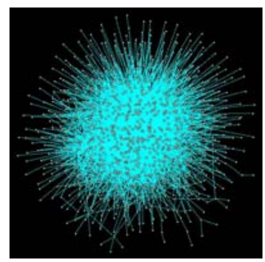
- Power-law properties verified ("find everything close by")
- Backbone + outskirts

Power-Law Random Graph (PLRG):

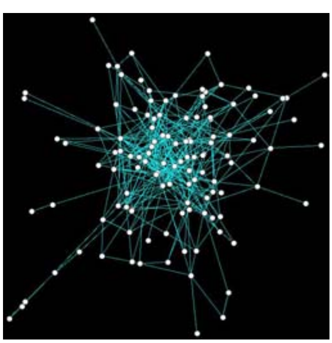
The node degrees follow a power law distribution:

if one ranks all nodes from the most connected to the least connected, then the i'th most connected node has ω/i^{a} neighbors,

where w is a constant.

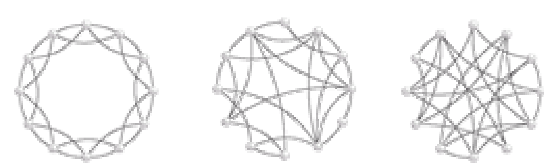


Gnutella Backbone [Jovanovic01]



Why does it work? It's a small World! [Hong01]

- Milgram: 42 out of 160 letters from Oregon to Boston (~ 6 hops)
- Watts: between order and randomness
  - short-distance clustering + long-distance shortcuts



Regular graph:
n nodes, k nearest neighbors
⇒ path length ~ n/2k
4096/16 = 256

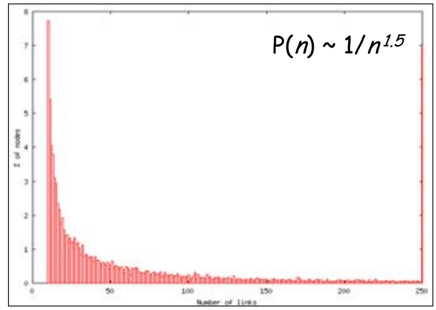
Rewired graph (1% of nodes):
path length ~ random graph
clustering ~ regular graph

Random graph:
path length ~ log (n)/log(k)
~ 4

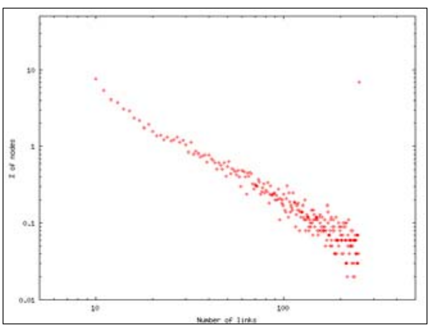
Links in the small World [Hong01]

- "Scale-free" link distribution
  - Scale-free: independent of the total number of nodes
  - Characteristic for small-world networks
  - The proportion of nodes having a given number of links *n* is:
    $P(n) = 1 / n^k$
  - Most nodes have only a few connections
  - Some have a lot of links: important for binding disparate regions together

Freenet: Links in the small World [Hong01]

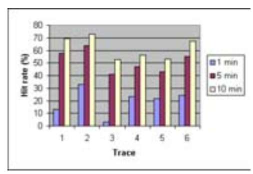


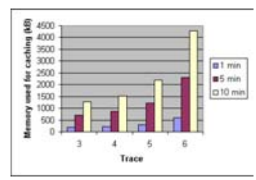
Freenet: "Scale-free" Link Distribution [Hong01]



Caching in Gnutella [Sripanidkulchai01]

- Average bandwidth consumption in tests: 3.5Mbps
- Best case: trace 2 (73% hit rate = 3.7 times traffic reduction)





Gnutella: New Measurements

[1] Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble:
A Measurement Study of Peer-to-Peer File Sharing Systems,
Proceedings of Multimedia Computing and Networking (MMCN)
2002, San Jose, CA, USA, January 2002.

[2] M. Ripeanu, I. Foster, and A. Iamnitchi.
Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design.
IEEE Internet Computing Journal, 6(1), 2002

[3] Evangelos P. Markatos,
Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella,
2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.

[4] Y. HawatheAWATHE, S. Ratnasamy, L. Breslau, and S. Shenker.
Making Gnutella-like P2P Systems Scalable. In Proc. ACM SIGCOMM (Aug. 2003).

[5] Qin Lv, Pei Cao, Edith Cohen, Kai Li, Scott Shenker:
***Search and replication in unstructured peer-to-peer networks**. ICS 2002: 84-95*

Gnutella: Bandwidth Barriers

- Clip2 measured Gnutella over 1 month:
  - typical query is 560 bits long (including TCP/IP headers)
  - 25% of the traffic are queries, 50% pings, 25% other
  - on average each peer seems to have 3 other peers actively connected
- Clip2 found a scalability barrier with substantial performance degradation if queries/sec > 10:

  10 queries/sec
  * 560 bits/query
  * 4 (to account for the other 3 quarters of message traffic)
  * 3 simultaneous connections
  67,200 bps
  ⇒ 10 queries/sec maximum in the presence of many dialup users
  ⇒ won't improve (more bandwidth - larger files)

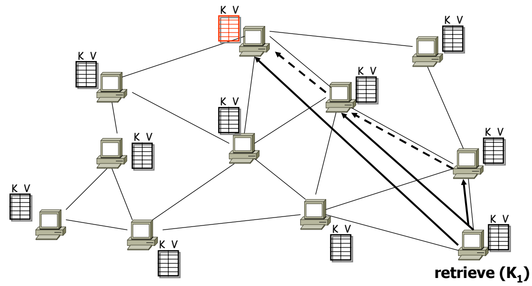
Gnutella: Summary

- Completely decentralized
- Hit rates are high
- High fault tolerance
- Adopts well and dynamically to changing peer populations
- Protocol causes high network traffic (e.g., 3.5Mbps). For example:
  - 4 connections *C* / peer, TTL = 7
  - 1 ping packet can cause packets $2*\sum_{i=0}^{TTL} C*(C-1)^i = 26{,}240$
- No estimates on the duration of queries can be given
- No probability for successful queries can be given
- Topology is unknown ⇒ algorithms cannot exploit it
- Free riding is a problem
- Reputation of peers is not addressed
- Simple, robust, and scalable (at the moment)

Iterative vs. Recursive Routing

Iterative: Originator requests IP address of each hop
- Message transport is actually done via direct IP

Recursive: Message transferred hop-by-hop



Hierarchical Networks (& Queries)

- DNS
  - Hierarchical name space ("clients" + hierarchy of servers)
  - Hierarchical routing w/aggressive caching
    - 13 managed "root servers"
- Traditional pros/cons of Hierarchical data mgmt
  - Works well for things aligned with the hierarchy
    - Esp. physical locality
  - Inflexible
    - No data independence!

Commercial Offerings

- JXTA
  - Java/XML Framework for p2p applications
  - Name resolution and routing is done with floods & superpeers
    - Can always add your own if you like
- MS WinXP p2p networking
  - An unstructured overlay, flooded publication and caching
  - "does not yet support distributed searches"
- Both have some security support
  - Authentication via signatures (assumes a trusted authority)
  - Encryption of traffic

Lessons and Limitations

- Client-Server performs well
  - But not always feasible
    - Ideal performance is often not the key issue!
- Things that flood-based systems do well
  - Organic scaling
  - Decentralization of visibility and liability
  - Finding popular stuff (e.g., caching)
  - Fancy *local* queries
- Things that flood-based systems do poorly
  - Finding unpopular stuff [Loo, et al VLDB 04]
  - Fancy *distributed* queries
  - Vulnerabilities: data poisoning, tracking, etc.
  - Guarantees about anything (answer quality, privacy, etc.)

Summary and Comparison of Approaches

	Paradigm	Search Type	Search Cost (messages)	Autonomy
Gnutella	Breadth-first search on graph	String comparison	$2*\sum_{i=0}^{TTL} C*(C-1)^i$	very high
FreeNet	Depth-first search on graph	String comparison	O(Log n) ?	very high
Chord	Implicit binary search trees	Equality	O(Log n)	restricted
CAN	d-dimensional space	Equality	O(d n^(1/d))	high
P-Grid	Binary prefix trees	Prefix	O(Log n)	high

More on Search

Search Options
- Query Expressiveness (type of queries)
- Comprehensiveness (all or just the first (or k) results
- Topology
- Data Placement
- Message Routing

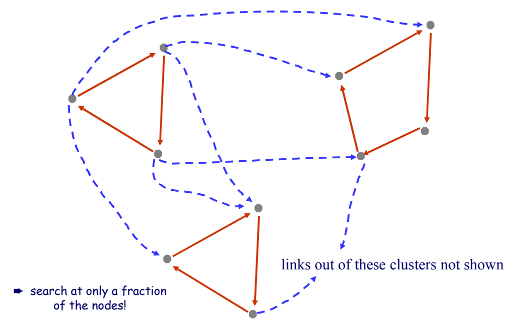
Comparison

	Gnutella	CAN	Others?
Expressivness	****		
Comprehensivness	**		
Autonomy	****		
Efficiency	*		
Robustness	***		
Topology	pwr law		
Data Placement	arbitrary		
Message Routing	flooding		

Comparison

	Gnutella	CAN	Others?
Expressivness	****	*	
Comprehensivness	**	****	
Autonomy	****	**	
Efficiency	*	***	
Robustness	***	**	
Topology	pwr law	grid	
Data Placement	arbitrary	hashing	
Message Routing	flooding	directed	

Parallel Clusters



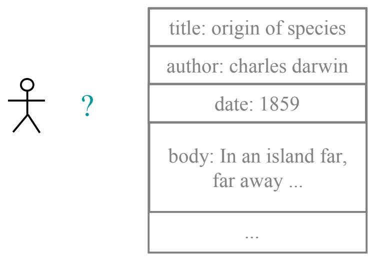
Other Open Problems besides Search: Security

- Availability (e.g., coping with DOS attacks)
- Authenticity
- Anonymity
- Access Control (e.g., IP protection, payments,...)

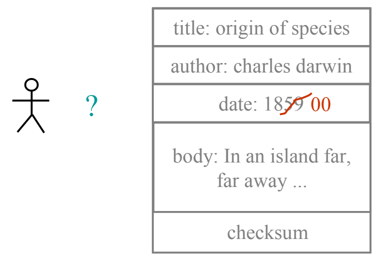
Trustworthy P2P

- Many challenges here. Examples:
  - Authenticating peers
  - Authenticating/validating data
    - Stored (poisoning) and in flight
  - Ensuring communication
  - Validating distributed computations
  - Avoiding Denial of Service
    - Ensuring fair resource/work allocation
  - Ensuring privacy of messages
    - Content, quantity, source, destination

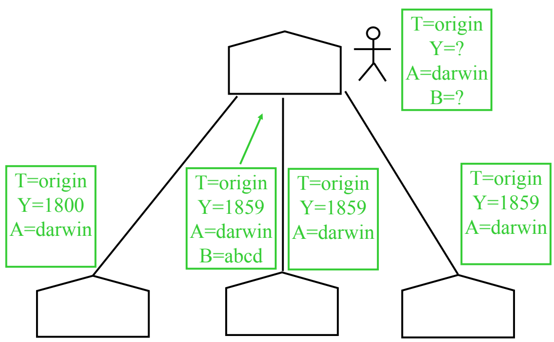
Authenticity



More than Just File Integrity



More than Fetching One File



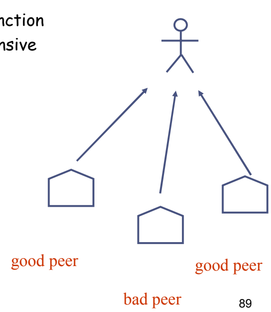
Solutions

- Authenticity Function A(doc): T or F
  - at expert sites, at all sites?
  - can use signature expert sig(doc) user
- Voting Based
  - authentic is what majority says
- Time Based
  - e.g., oldest version (available) is authentic

Added Challenge: Efficiency

- Example: Current music sharing
  - everyone has authenticity function
  - but downloading files is expensive

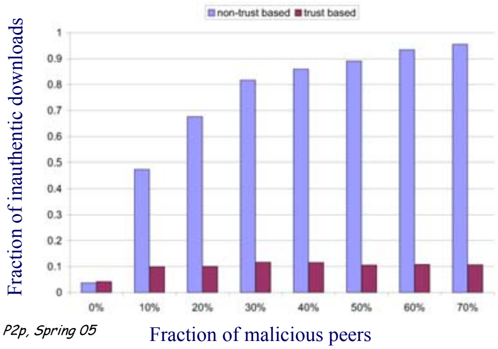
- Solution: Track peer behavior



Issues

- Trust computations in dynamic system
- Overloading good nodes
- Bad nodes can provide good content sometimes
- Bad nodes can build up reputation
- Bad nodes can form collectives
- ...

Sample Results



Security & Privacy

- Issues:
  - Anonymity
  - Reputation
  - Accountability
  - Information Preservation
  - Information Quality
  - Trust
  - Denial of service attacks

P2P Challenges

- Search ✔
- Resource Management
- Security & Privacy ✔

DAMD P2P!

- Distributed Algorithmic Mechanism Design (DAMD)
  - A natural approach for P2P
- An Example: Fair-share storage [Ngan, et al., Fudico04]
  - Every node *n* maintains a *usage record:*
    - Advertised capacity
    - *Hosted list* of objects n is hosting (nodeID, objID)
    - *Published list* of objects people host for *n* (nodeID, objID)
  - Can publish if capacity - $p \cdot \Sigma$(published list) > 0
    - Recipient of publish request should check *n*'s usage record
  - Need schemes to authenticate/validate usage records
    - Selfish Audits: *n* periodically checks that the elements of its hosted list appear in published lists of publishers
    - Random Audits: *n* periodically picks a peer and checks all its hosted list items

Lessons and Limitations

- Client-Server performs well
  - But not always feasible
    - Ideal performance is often not the key issue!
- Things that flood-based systems do well
  - Organic scaling
  - Decentralization of visibility and liability
  - Finding popular stuff (e.g., caching)
  - Fancy *local* queries
- Things that flood-based systems do poorly
  - Finding unpopular stuff [Loo, et al VLDB 04]
  - Fancy *distributed* queries
  - Vulnerabilities: data poisoning, tracking, etc.
  - Guarantees about anything (answer quality, privacy, etc.)