

P2P File Sharing Networks & Distributed Hash Tables (DHTs)

莊 裕 澤

Yuh-Jzer Joung

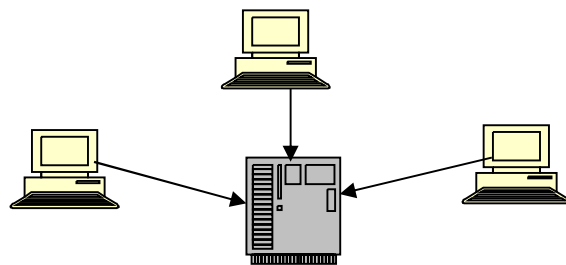
Dept. of Information Management
National Taiwan University

Outline

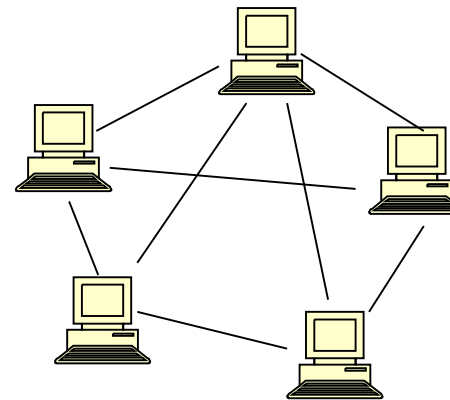
- ❑ **Part I: Unstructured P2P Networks**
 - Gnutella
 - Freenet
 - BitTorrent
- ❑ **Part II: Structured P2P Networks & DHTs**

What is a Peer-to-Peer Network?

- ❑ A network in which all nodes cooperate with each other to provide specific functionality with minimal centralized control
 - No distinction between client and server.
 - All nodes are potential users of a service *and* potential providers of a service.
- ❑ The motivation is to share resources, which could be files, storage, CPU cycles, bandwidth, ...etc



Client/Server

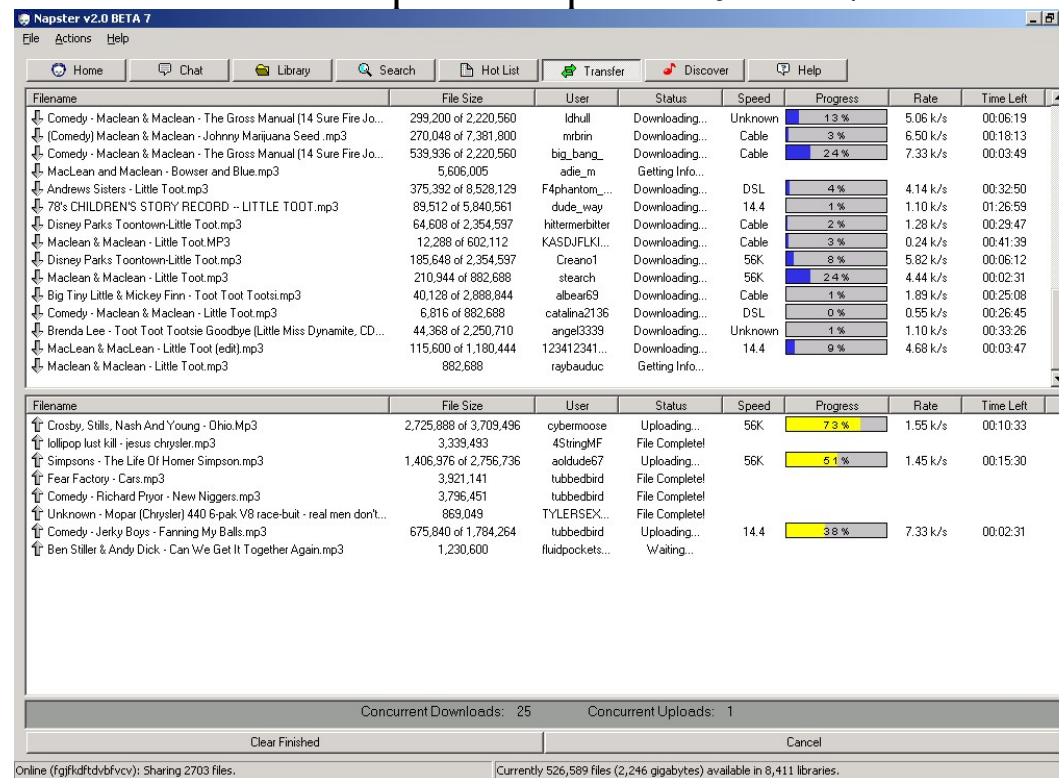


Peer-to-Peer

Milestone: Napster (1999.06-2001.07)

- ❑ Napster: pioneer of online music file sharing service
 - huge impact to the music industry
 - paved the way for decentralized peer-to-peer file-distribution programs.

The old Napster: Napster 2.0 Beta 7 user



The Epic of Napster

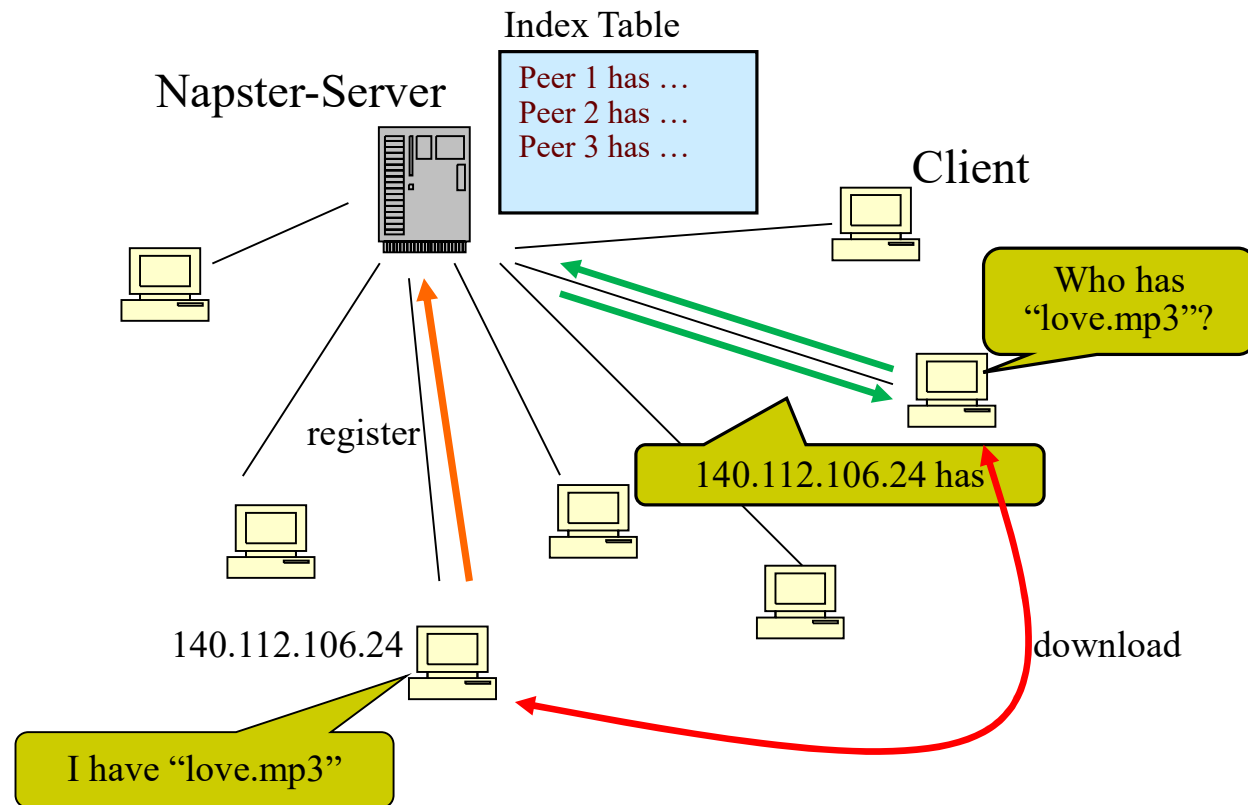


- ❑ Founded in June 1999 by Shawn Fanning, a 19-year-old Northeastern University dropout
 - First massively popular peer-to-peer file sharing systems
- ❑ Immensely successful
 - Success due to ability to create and foster an online community
 - Had 640,000 users at any given moment in November 2000
 - More than 60 million people, including an estimated 73% of US students, had downloaded the software and songs in less than one year.
 - At the peak, 1 million new users per week
 - Universities begin to ban Napster, due to overwhelmed bandwidth consumption
- ❑ Battled with RIAA
 - RIAA sued Napster in Dec. 1999, asking \$100K per song copied
 - Court ruled Napster to shutdown in July 2001
 - Filed for bankruptcy in June 2002



Shawn Fanning, 1980.11-

Napster Architecture



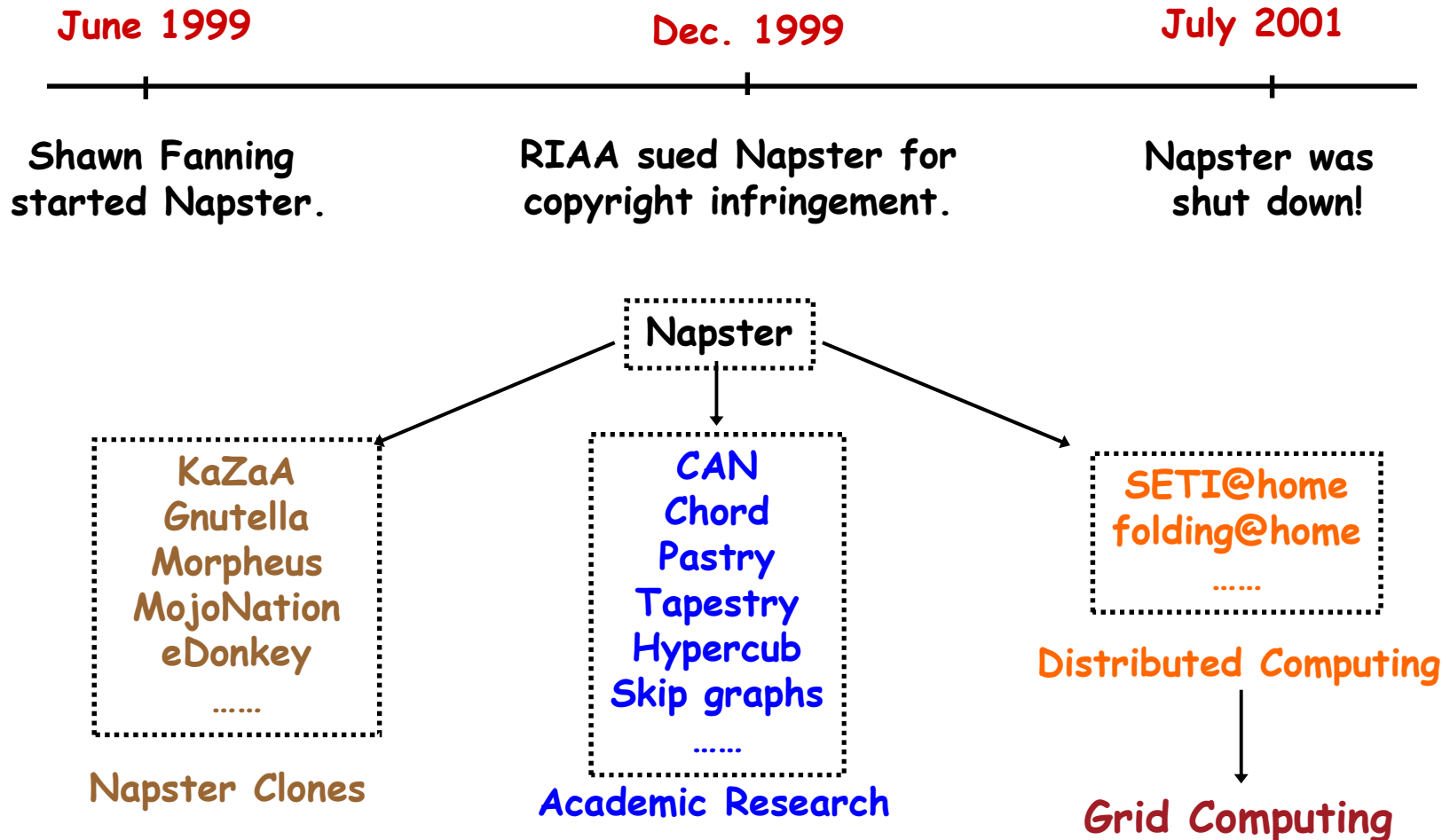
- ❑ Centralized design, with bottlenecks and vulnerability
- ❑ Simple implementation
- ❑ Low scalability

Napster: Reborn

- ❑ In May, 2002, German media firm Bertelsmann offered \$8M to acquire Napster, but was blocked by an American bankruptcy judge, forcing Napster to liquidate its assets
- ❑ In Sep. 2002, porn giant Private Media Group made a \$2.4M offer to acquire Napster's name and logo to help entering the P2P marketplace for adult content
- ❑ In Nov. 2002, Roxio, a company which makes software for CD-burners, bought Napster's name and associated patents for \$5M at bankruptcy auction
- ❑ In Oct. 2003, it became the second legal music download service to gain widespread popularity, 6 months after iTunes.
- ❑ Revenue \$94.69M in 2006, \$111.1M in 2007.
- ❑ Purchased by Best Buy for \$US 121M in Sep 2008.
- ❑ merged with Rhapsody in 2011.12



Napster Has Inspired a Flurry of Research on P2P



Seti@home

Search for Extraterrestrial Intelligence

❑ Purpose:

- Released in May 1999 by UC Berkeley to use idle CPU cycles on volunteers' PCs for massively parallel analysis of extraterrestrial radio signals

❑ Architecture:

- central SETI@home server distributes data, similar to Napster
- analyses done locally by SETI@home screen saver

❑ Statistics

- according to a statistics in 2004 ([see update](#))
 - Nearly 5M participants in 226 countries
 - Nearly 2M CPU years of work
 - Over 1.3B results received
- stopped sending out new work to SETI@home users in [2020.03](#)



P2P & DHTs



Berkeley Open Infrastructure for Network Computing (BOINC)

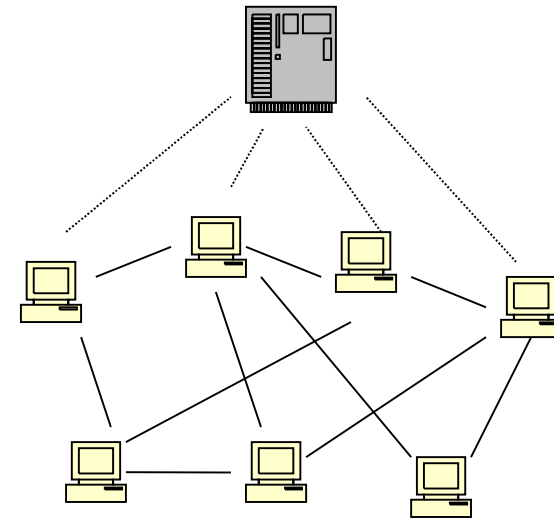
- ❑ SETI@home has evolved into BOINC (2002.04-), a general distributed computing platform based on volunteers' computing power to host several computation-demanding projects, e.g., scientific modeling, protein folding, earth science, medicine, physics...
 - Computing power (as of 2010.06.06, see [update](#)):
 - 1.97M users, 315K of them active
 - 5.2M hosts, 565K of them active
 - Average computing power: 5.4 peta FLOPS
 - the fastest supercomputer system
 - [\(as of 2010.06\)](#)
 - Cray XT5 (Jaguar)
 - 1.75PFLOPS



Cray XT5 Jaguar at Oak Ridge National Laboratory

Classification of P2P

- ❑ Centralized Systems
 - central server to assist administration
 - Napster, Seti@Home
- ❑ Fully Decentralized Systems
 - Unstructured:
 - Gnutella, Freenet
 - Structured
 - CFS/Chord, Tapestry, Pastry
- ❑ Hybrid Systems
 - Morpheus, KaZaA, eDonkey



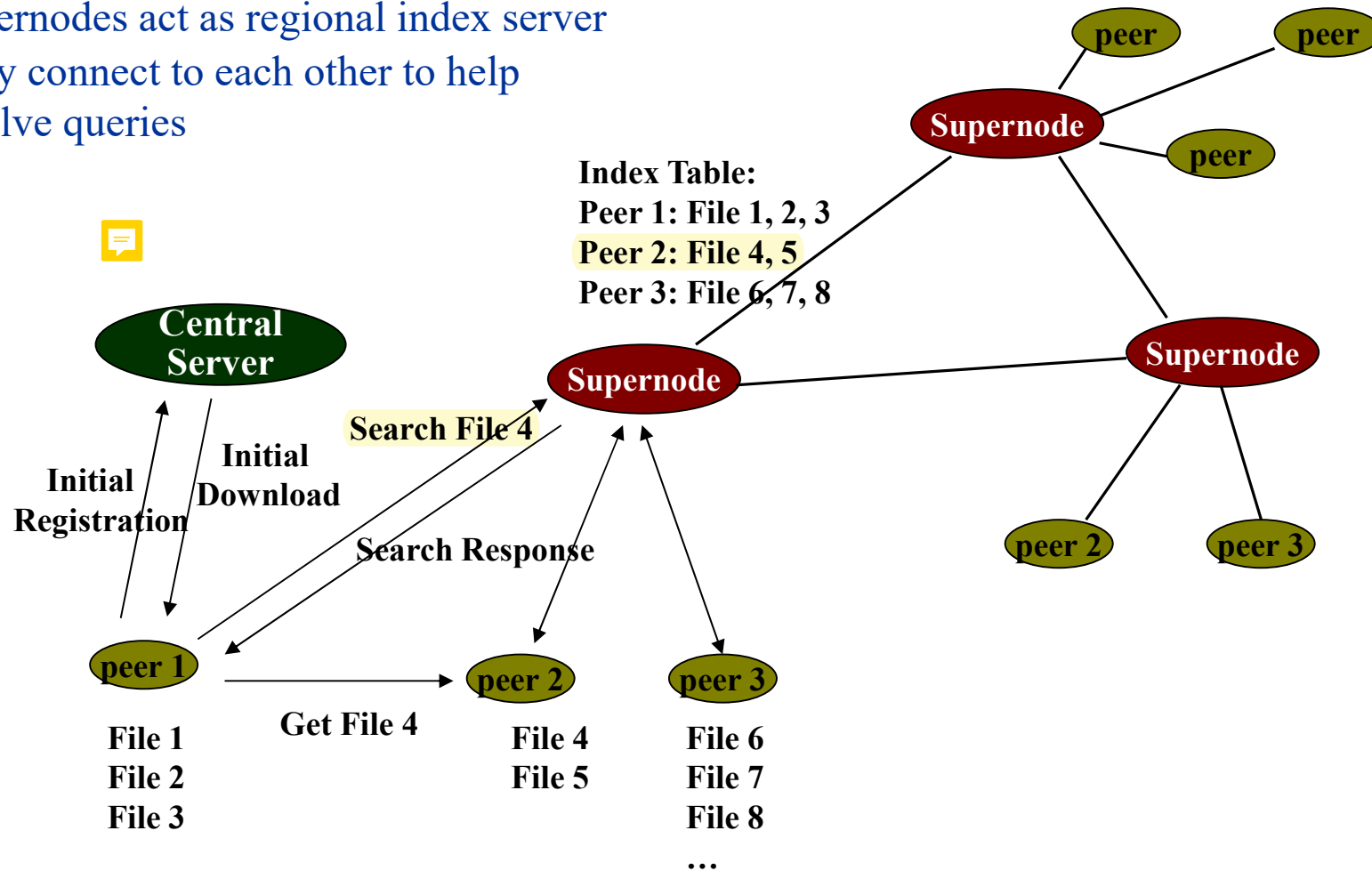
Hybrid Structure

- ❑ Has a centralized server that maintains user registrations, logs users into the systems to keep statistics, provides downloads of client software, and bootstraps the peer discovery process.
- ❑ Two types of peers:
 - Supernodes (fast CPU + high bandwidth connections)
 - Nodes (slower CPU and/or connections)
- ❑ Supernode addresses are provided in the initial download. They also maintain searchable indexes and proxies search requests for users. A node may connect to more than one supernodes, and may change the list dynamically.
- ❑ Examples: KaZaA, eMule, Morpheus



Super Peer Architecture

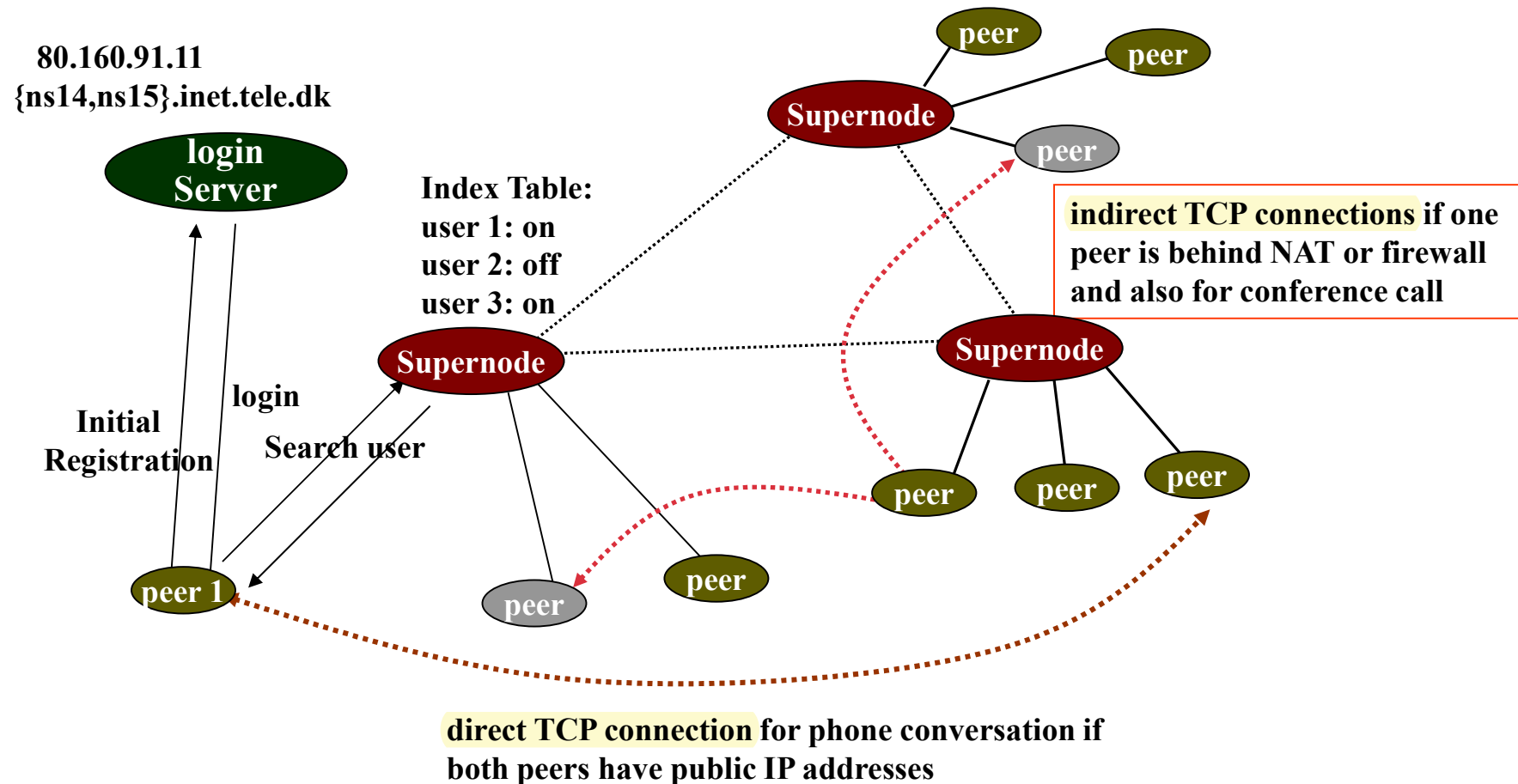
- ❑ Supernodes act as regional index server
- ❑ They connect to each other to help resolve queries





Skype Architecture (ca. 2003-2016)

- Skype was developed in 2003 by the same people who developed KaZaa
- Used a hybrid structure with super nodes to store online and offline user information and to offer most of the services



Skyped acquired by eBay and Microsoft

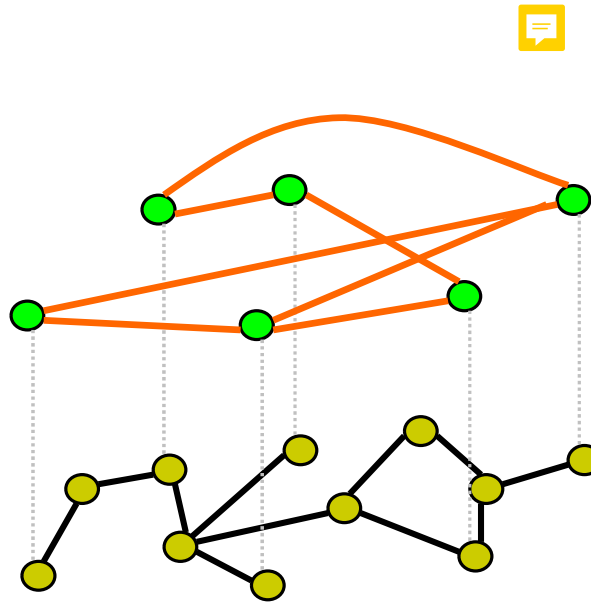
- ❑ eBay acquires Luxembourg-based Skype in \$4.1 billion on Sep. 12, 2005.
 - \$1.3b in cash and \$1.3 billion in stock and a further payout of up to \$1.5 billion if certain financial targets are met (WashingtonPost, 2005.09.13)
 - P.s. eBay bought PayPal for \$1.5B in July 2002
 - In Oct. 2007 eBay wrote off \$1.43B of its investment in Skype, admitting that it “drastically overpaid” for the company. (NY Times)
 - In Nov. 2009, eBay sold 70% of Skype to a group of investors, valued it at \$2.75B (just slightly higher than the \$2.6B eBay paid to acquire Skype) (TechCrunch 2009.11.19).
- ❑ Microsoft bought Skype for \$8.5b, 2011.05.10




Fully Decentralized Unstructured P2P Networks

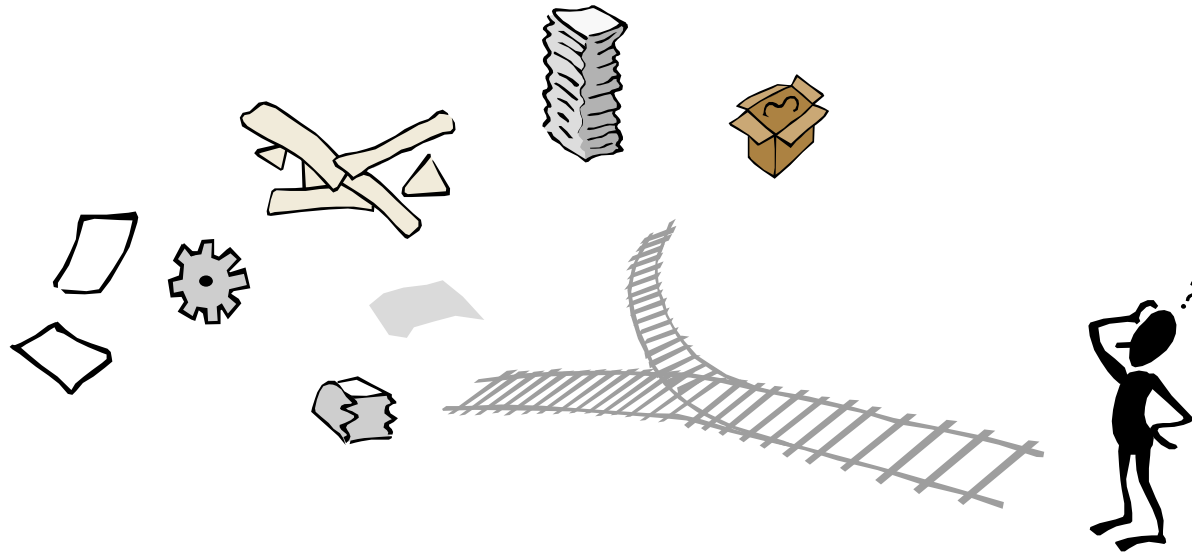
Overlay

- ❑ Fully distributed P2P Networks are essentially an *overlay* over the Internet---an application layer network.
 - How to select neighbors to build an overlay?



Main Technical Challenge in P2P: Locating the Target

- ❑ How to find needed information/object given that
 - Dynamics: nodes can join and leave any time
 - Scalability: there may be millions of hosts 



The problem is trivial in a Napster-like P2P system with a centralized server to manage the index

Gnutella



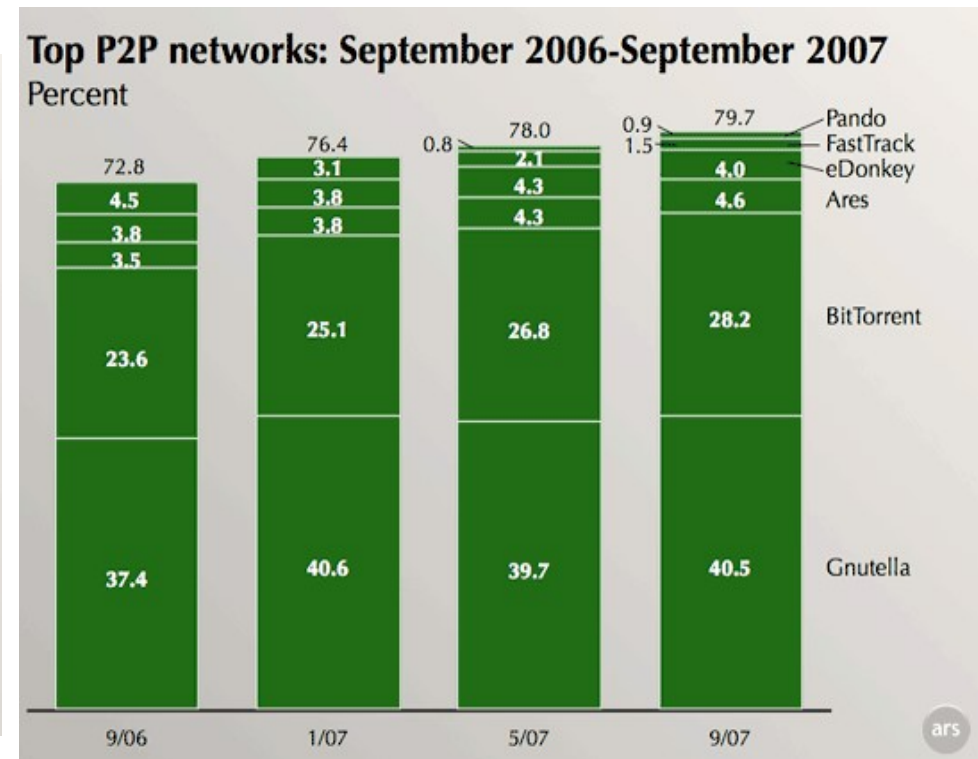
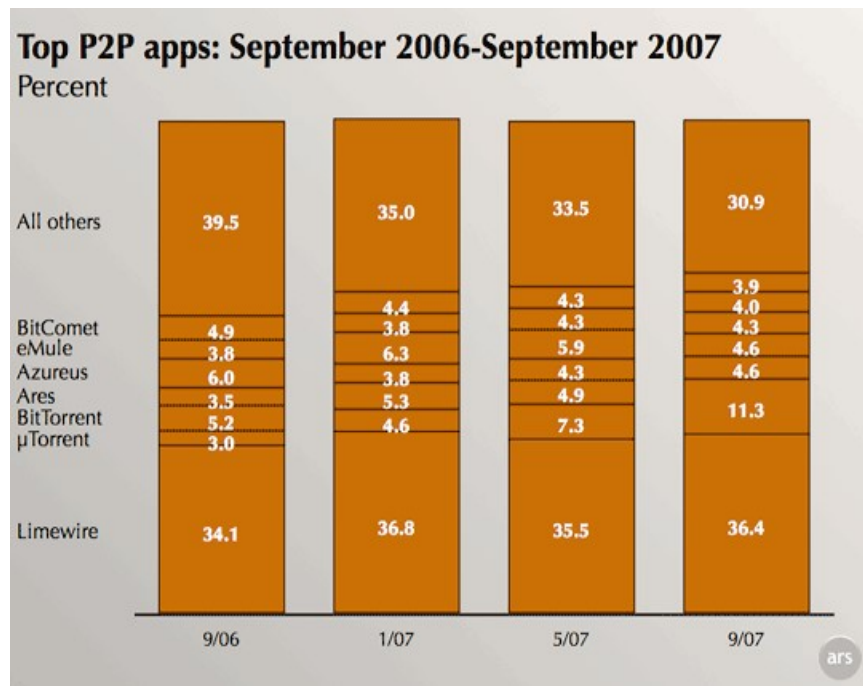
- ❑ Developed by Justin Frankel (and Tom Pepper) of Nullsoft in early 2000 by Justin Frankel, then 21 year old, founder of Nullsoft (which was acquired by America Online (AOL) in Jun 1999), and a University of Utah dropout.
- ❑ The program was made available for download on Nullsoft's servers on 2000.03.14.
- ❑ A day later AOL stopped the availability of the program over legal concerns and restrained Nullsoft from doing any further work on the project (because AOL was at the time merging with Time Warner).
- ❑ However, the application had already distributed as open source, and different versions started to develop
 - Limewire, Gnucleus, Morpheus
 - It is now more like a protocol



Justin Frankel (source: CNN)

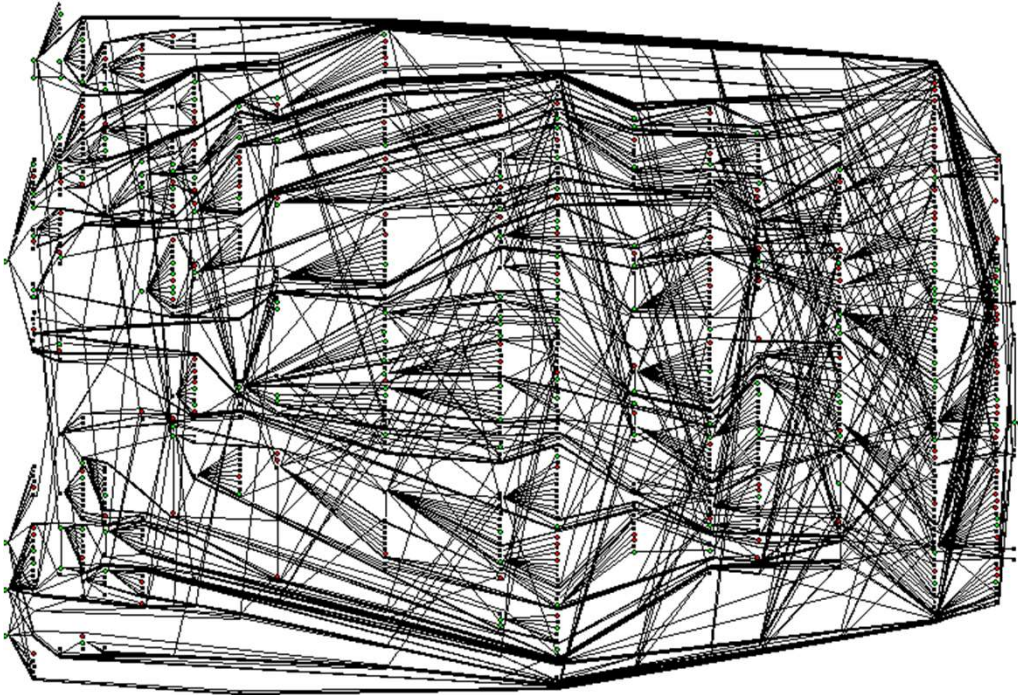
Gnutella Population in 2000s

- In 2007, it has about 40% of the P2P client market share, and an estimated 3M+ clients in 2006 (in terms of traffic, BT is No.1 (45-78% of all P2P traffic, and 27-55% of all Internet traffic as of 2009) as it is more suitable for movies, while Gnutella is usually for music.)

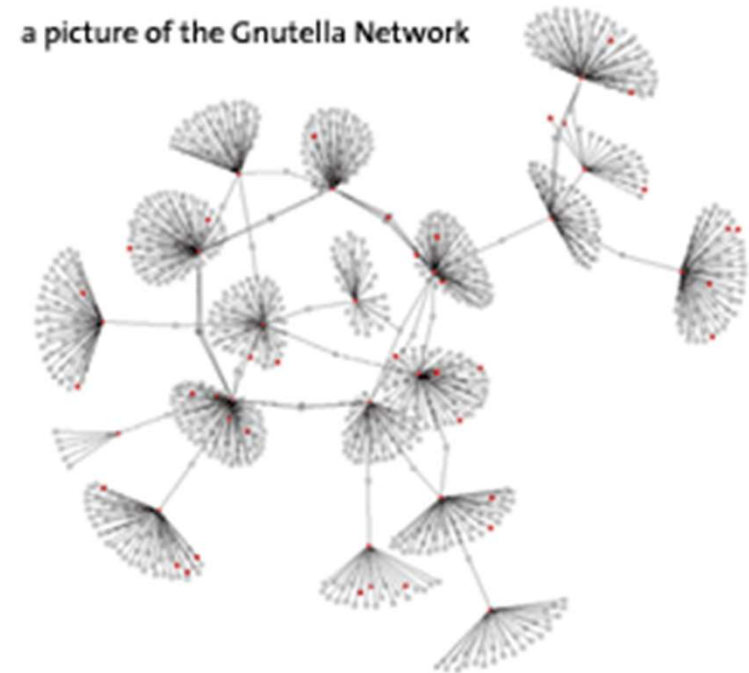


ARS, 2008.04.21

Partial Map of the Gnutella Network (ca. 2000)



Source: <http://dss.clip2.com>



Source: <http://www.limewire.com>

Gnutella Overlay Construction

❑ Forming the overlay

- node join: connect to several known hosts



- node leave:

- benign: inform neighboring nodes
- silent

❑ Maintaining the overlay

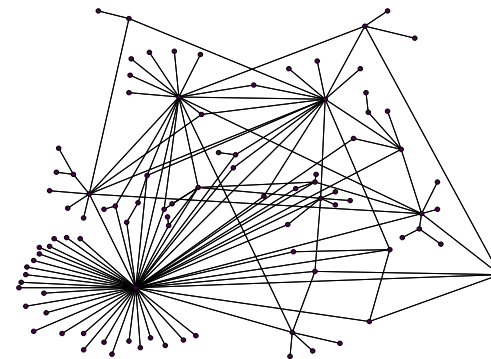
- add/update entries in routing tables

- There are several heuristics to select neighbors

- Think of how you maintain your phonebook in the IM network

❑ Node discovery and search

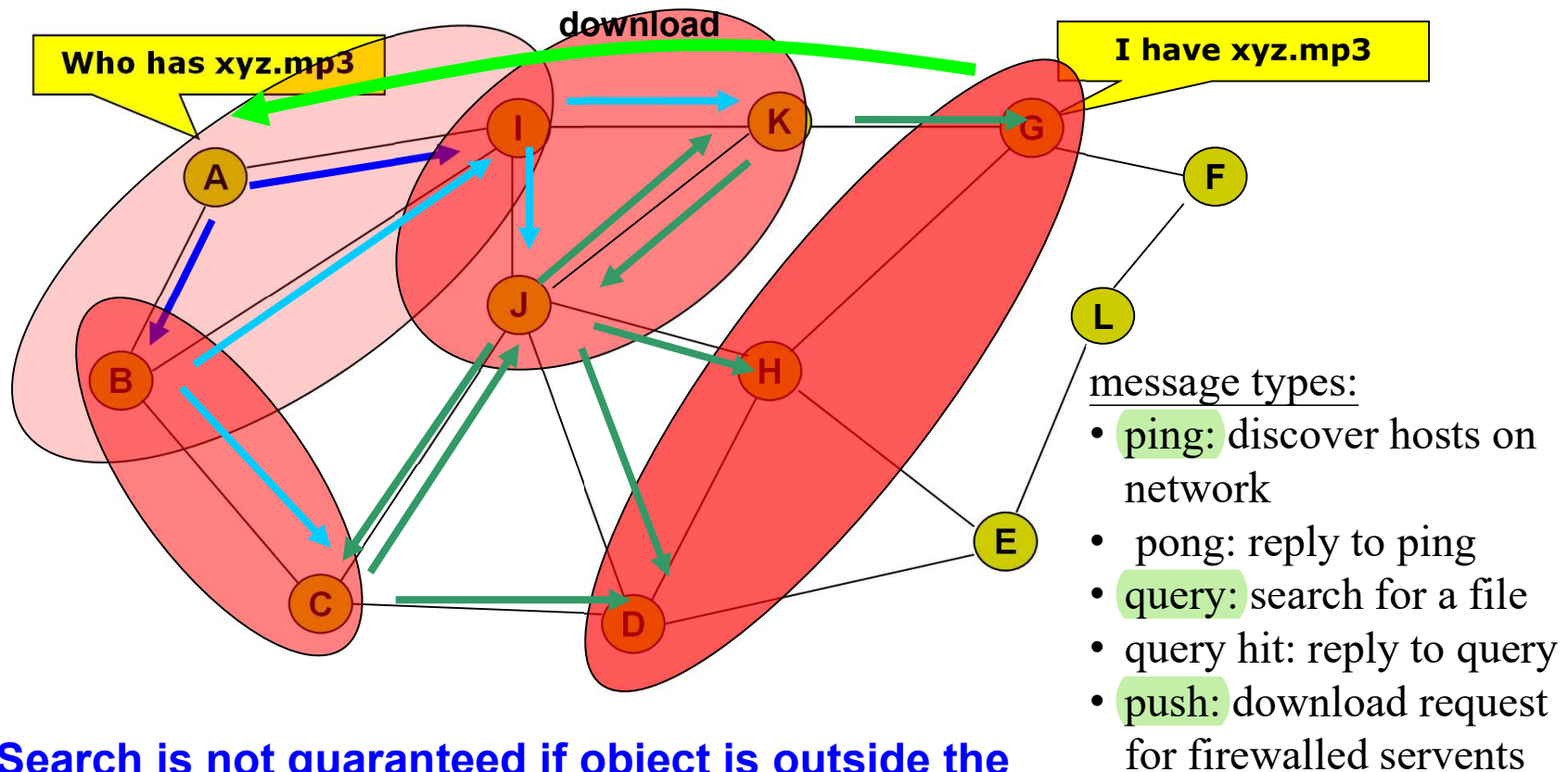
- by broadcast



Search in Gnutella



Basically flooding/breadth-first search



Search is not guaranteed if object is outside the search range!

Gnutella Network Analysis

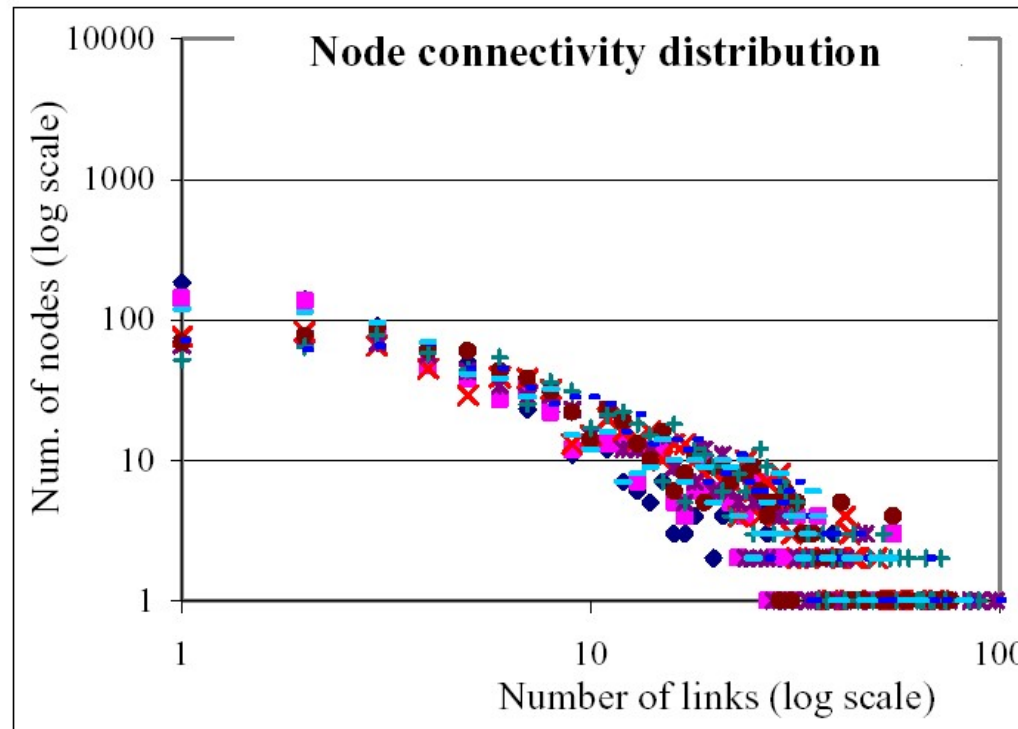


Figure 5: Connectivity distribution during November 2000. Each series of points represents one Gnutella network topology we discovered at different times during that month. Note the log scale on both axes. Gnutella nodes organized themselves into a **power-law network**.

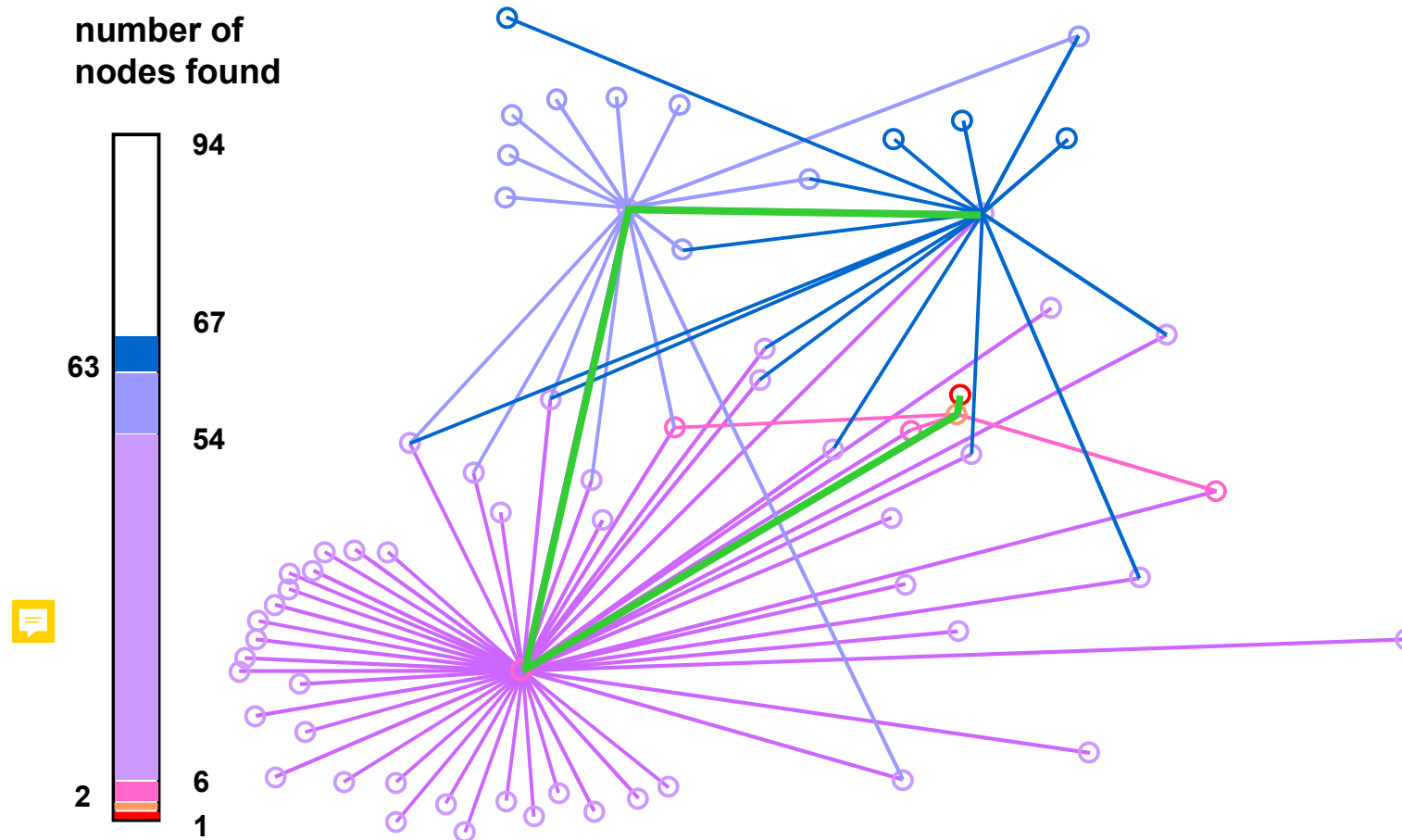


Reference: Peer-to-Peer Architecture Case Study: Gnutella Network, Matei Ripeanu, in proceedings of IEEE 1st Int'l Conf. on Peer-to-peer Computing, 2001 (research conducted in 2000-2001).

Power-Law Networks

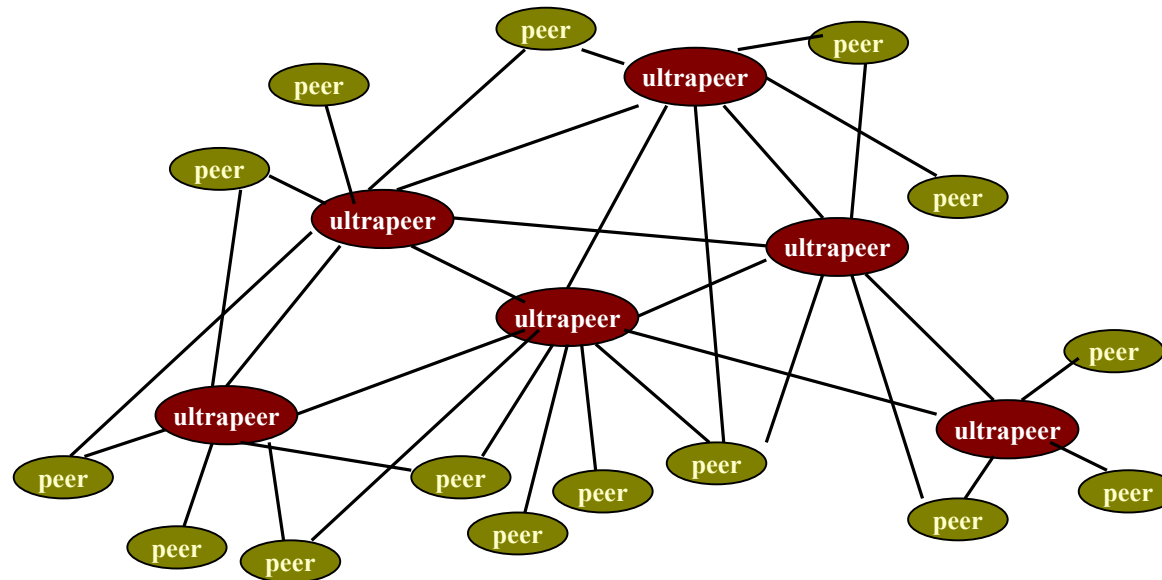
- ❑ In a **power-law network**, the fraction of nodes with L links is proportional to L^{-k} , where k is a network dependent constant.
 - most nodes have few links and a tiny number of hubs have a large number of links.
- ❑ Power-law networks are generally highly stable and resilient, yet prone to occasional catastrophic collapse.
 - extremely robust when facing random node failures, but vulnerable to well-planned attacks.
- ❑ The power law distribution appears in Gnutella networks, in Internet routers, in the Web, in call graphs, ecosystems, as well as in sociology.
- ❑ Related terms: Zipf law, Pareto law (aka the 80-20 rule, the law of the vital few and the principle of factor sparsity).

Power-Law Graph



Evolution of Gnutella

- ❑ Newer versions of Gnutella (v.0.6 and Gnutella 2) have adopted an **ultrapeer** (super peer) structure. 💬
 - Leaf nodes connect to some (typically 3) ultrapeers, while ultrapeer connects to more than 32 other ultrapeers.
 - Ultrapeers can also be tiered, resembling the DNS hierarchical structure.



Gnutella Summary

- ❑ Ease of construction and maintenance
- ❑ Robust and resilient to the dynamics of the network
- ❑ Flexibility in query format
 - exact name match
 - keyword match
 - wildcard search: “*.mp3”, “lord ring *”
 - range queries: “files created in 2000-2010”
 - complex search
- ❑ Low scalability
 - huge bandwidth consumption
 - e.g. TTL=7, every node broadcasts to 5 others
 - ✓ total of 78125 messages
- ❑ Search cannot be guaranteed!
 - low TTL, low search horizon

Freenet

- ❑ A project led by Ian Clarke in his 4th Year Project at University of Edinburgh (1999)
- ❑ Philosophy
 - One should be able to publish and obtain information on the Internet without fear of censorship
- ❑ The result
 - A Distributed Anonymous Information Storage and Retrieval System



Reading:

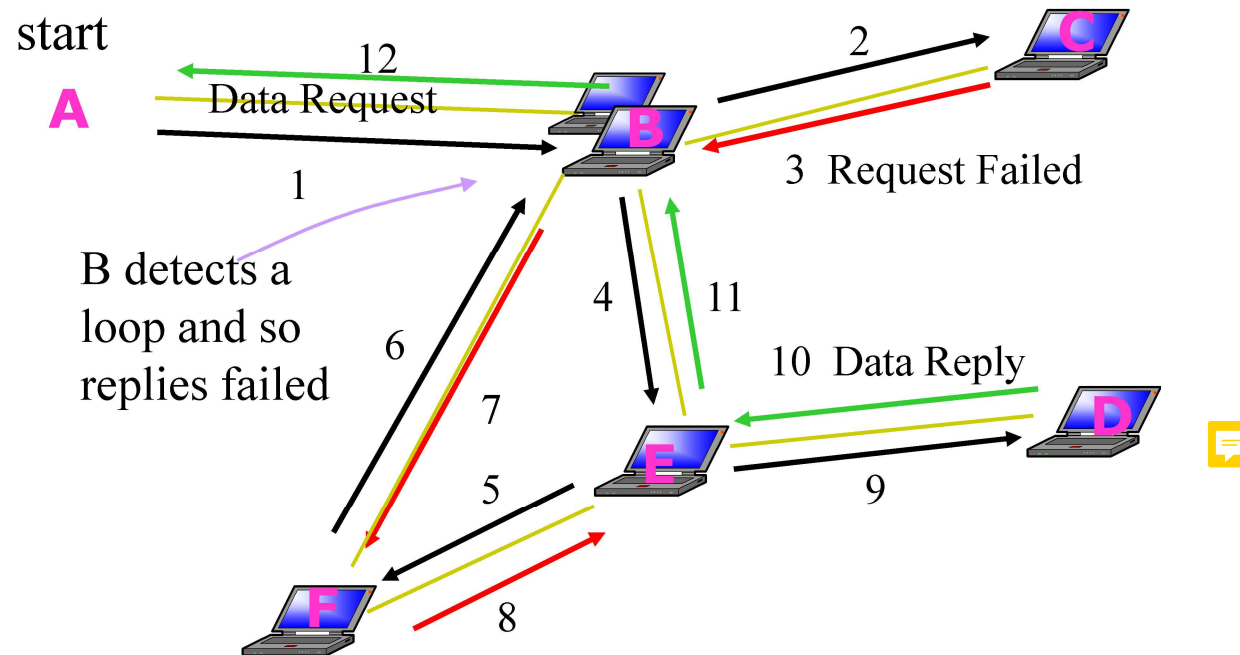
Freenet: A Distributed Anonymous Information Storage and Retrieval System. Ian Clarke, et al., Springer 2001.

Design Goals



- ❑ Anonymity for both producers and consumers of information
- ❑ Deniability for stores of information
- ❑ Resistance to attempts by third parties to deny access to information
- ❑ Efficient dynamic storage and routing of information
- ❑ Decentralization of all network functions

Protocol Detail: Retrieving Data



Basically a **depth-first search (DFS)**!

DFS in general is much **less efficient** than Gnutella's BFS.

How can it be made more efficient?

Neighbor Selection in Routing

- ❑ A description of the file to be requested (e.g., its file name) is hashed to obtain a file key that is attached in the request message.
- ❑ A node receiving a request for a file key checks if its store has the file. If so, it returns the file. Otherwise, it forwards the request to the node in its routing table that has the most 'similar' key to the requested one.
- ❑ If that node cannot successfully (and recursively!) find the requested file, then a second (again, according to key similarity) candidate from the routing table is chosen, and so on, until either the file is found, or a request fail message is returned.
- ❑ Once a copy of the requested file is found, it is sent to the requester along the search path. Each node in the path caches a copy of the file, and creates a new entry in its routing table associating the actual data source with the requested key (*path replication*).

Some Observations (I)

- ❑ Quality of routing should improve over time
 - Nodes should come to specialize in locating similar keys.
 - Nodes should become similarly specialized in storing files having similar keys.
- ❑ Popular data will be replicated and mirrored closer to requesters.
- ❑ Connectivity increases as requests are processed.
- ❑ Note that files with similar hashed keys do not mean that they are similar in content.
 - A crucial node failure cannot cause a particular subject to extinguish.

File Insert

- ❑ Insert basically follows the same depth-first like search to see if there is a key collision. If so, the colliding node returns the pre-existing file as if a request has been made. If no collision, then the file will be placed on each node in the search path.
 - The nodes will also update their routing tables, and associate the inserter as the data source with the new key.
 - For security reason, any node along the way can unilaterally decide to change the insert message and claim itself or another arbitrarily node as the data source.

Observations (II)

- ❑ Newly inserted files are placed on nodes already possessing files with similar keys.
- ❑ New nodes can use inserts as a supplementary means of announcing their existence to the rest of the network.
- ❑ An attempt to supply spurious files will likely to simply spread the real file further, as the original file is propagated back on collusion.

Anonymity in Freenet

❑ Achieved by the combination of path replication and the use of the following two counters:

- Hops-to-live

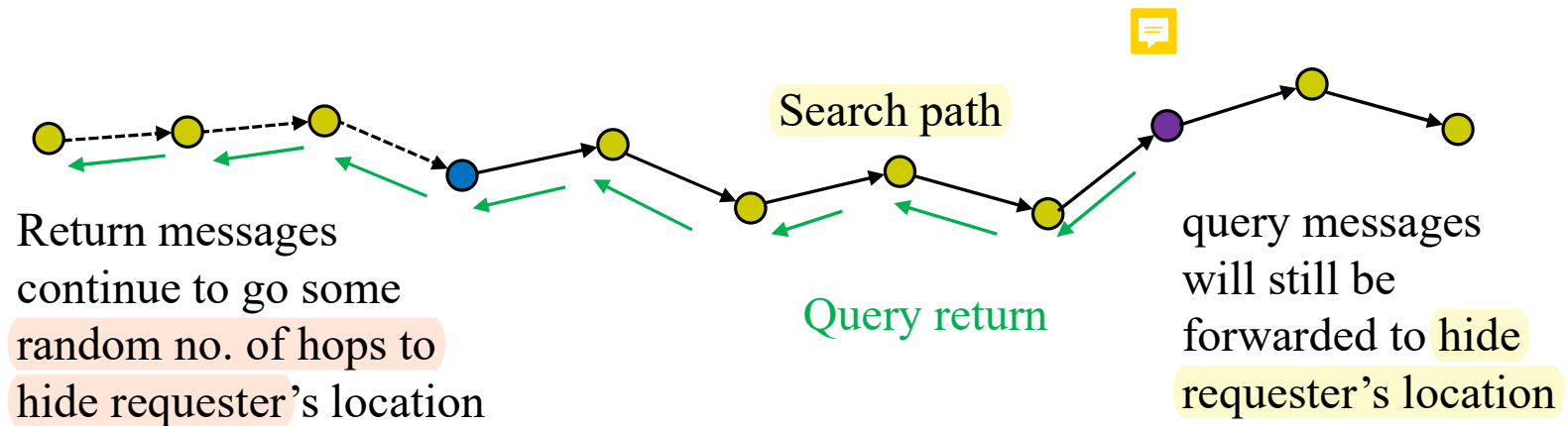
- Decrement at each hop to prevent indefinite routing
- Messages will still be forwarded with some probability even though hops-to-live value has reached 1, so as to prevent attacker from tracing the requester's location.

- Depth counter

- Incremented at each hop to let a replying node set hops-to-live high enough to reach a request.
- Requestors should initialize it to a small random value to obscure their location.

Why use two counters instead of one?

Anonymity in Freenet (2)



Managing Data

- ❑ Node storage is managed as an LRU (Least Recently Used) cache.
- ❑ **No file is eternal!** A file could disappear in the network if it has not been requested for a long period of time.
- ❑ Inserted files are encrypted so that node operators can 'claim' innocence of possessing some controversial files.
 - The requester who knows the actual file name can use that information to decrypt the encrypted file.

Freenet Naming

- ❑ Hierarchical name system
 - Files are identified by the hash of their filenames
 - Cannot have multiple files with the same name, thus global single-level namespace is not desirable
- ❑ Two - level namespace
 - Each user has their own directory

Freenet File Export

- ❑ Consider exporting file with name “My life.mp3”
 - Compute a public/private key pair from name using a deterministic algorithm
- ❑ File is encrypted with the hash of the public key
 - Goal is not to protect data – the file contents should be visible to anyone who knows the original keyword
 - Goal is to protect site operators – if a file is stored on your system, you have no way of decrypting its contents
- ❑ File is signed with the private key
 - Integrity check (though not a very strong one)

Freenet Directories

- ❑ Two - level directories
 - Users can create a signed-subspace
 - Akin to creating a top-level directory per user
 - Subspace creation involves generating a public/private key pair for the user
 - The user's public key is hashed, XORed and then rehashed with the file public key to yield the file encryption key
- ❑ For retrieval, you need to know the user's public key and the file's original name

Adding Nodes

- ❑ Node connecting to network must obtain existing node's address through out-of-band means
- ❑ Once connected, a new node message is propagated to randomly selected, connected nodes so existing nodes learn of new node's existence

Freenet Summary

❑ Advantages

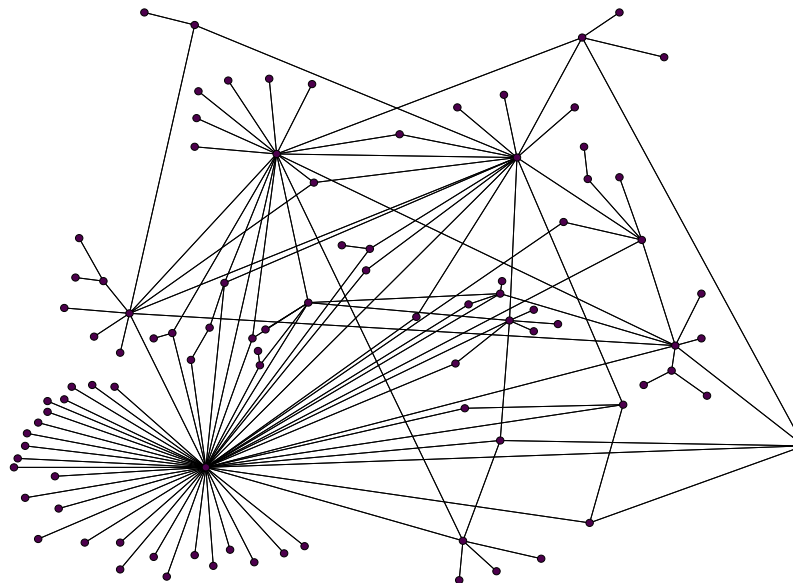
- Totally decentralize architecture
 - robust and scalable

❑ Disadvantages

- Does not always guarantee that a file is found, even if the file is in the network

Search Techniques in Unstructured P2P Networks

- ❑ Flooding (Breath-First Search, BFS)
- ❑ Depth-First Search
- ❑ Random Walk
- ❑ Search Enhancement



Random Walk

- Probability of search success rate for a particular node:

$$P = 1 - (1 - \frac{1}{N})^C$$

where N is the network size and C is the coverage.

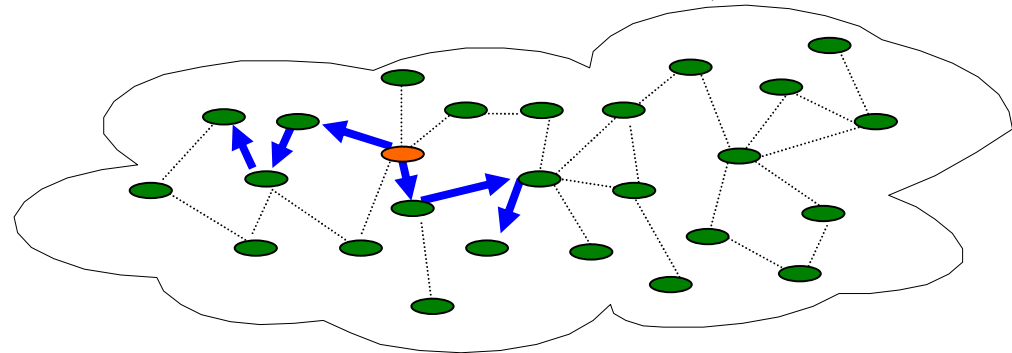
$C = w \times T$, w is the number of *walkers*, and T is the TTL of each walker.

- Let $C = \alpha N$, and assume there are r replicas of the target.

When N is large,

r	α	P
1	0.01	0.01
10	0.01	0.095
100	0.01	0.632
1000	0.01	0.9999

$$P = 1 - (1 - \frac{r}{N})^{\alpha N} \approx 1 - e^{-r\alpha}$$



Random walk is effective in searching popular objects.

Search Enhancement

❑ Iterative Deepening

first broadcasts a query message with the smallest depth. If the desired resource is not found, the technique increases the TTL and reissues the query

❑ Local Indices

each node maintains an index of files that are within some r hops.

❑ Interest-based locality

if a peer has a piece of content one is interested in, then it is likely to have other pieces of content that one is interested in as well.

❑ Directed Breadth-First Search

choosing among different neighbors one that may return the results to forward a query.

Replication

- ❑ Path replication:
 - a file is replicated along the return path from the source to the requesting node
 - used in Freenet, and has been proven to be quite effective in searching popular files.

Still...

- ❑ The above techniques help trimming the search tree of the node, but they have little help for finding a file that is far outside the search space.
- ❑ Similarly, good replication strategies help reducing the expected search space, especially for popular files; but unpopular files may still be out of the search range.

BitTorrent

BitTorrent: Brief History

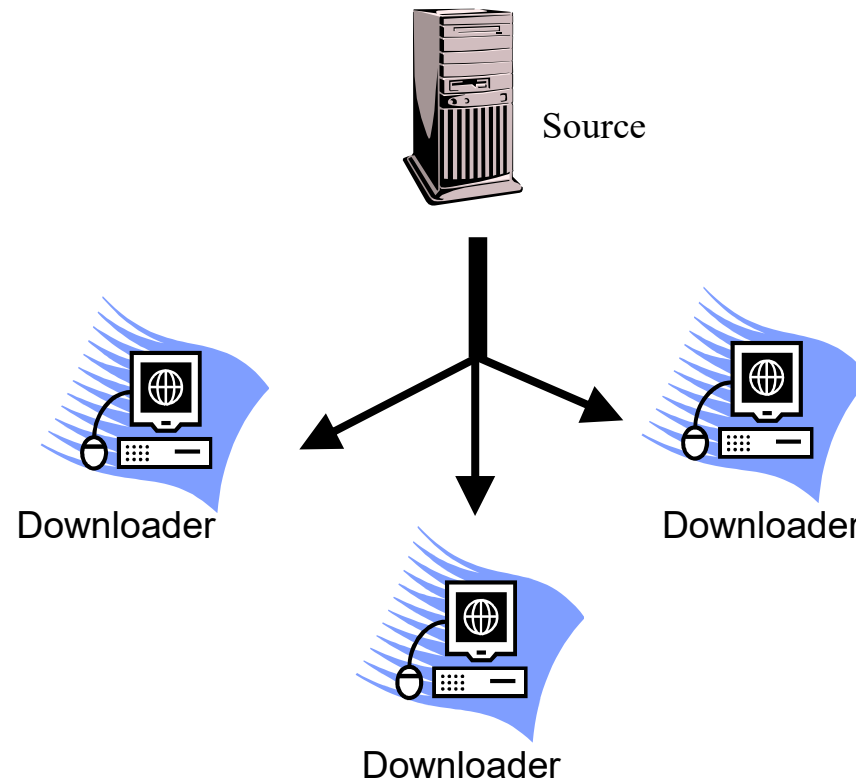
- ❑ Official Website : <http://bittorrent.com/>
- ❑ Created by Bram Cohen in 2001 (then age 26)
- ❑ Ideal for distributing large (gigabyte) files
- ❑ According to CacheLogic, BitTorrent accounts for 35% Internet traffic in Nov. 2004.
- ❑ Now maintained by [BitTorrent, Inc.](#) (founded in 2004), although there are numerous compatible BitTorrent clients (e.g., µTorrent, BitComet...)

Reading: [Incentives build robustness in bittorrent](#). B. Cohen., In Workshop on Economics of Peer-to-Peer systems, vol. 6, 2003.



Bram Cohen, 1975-

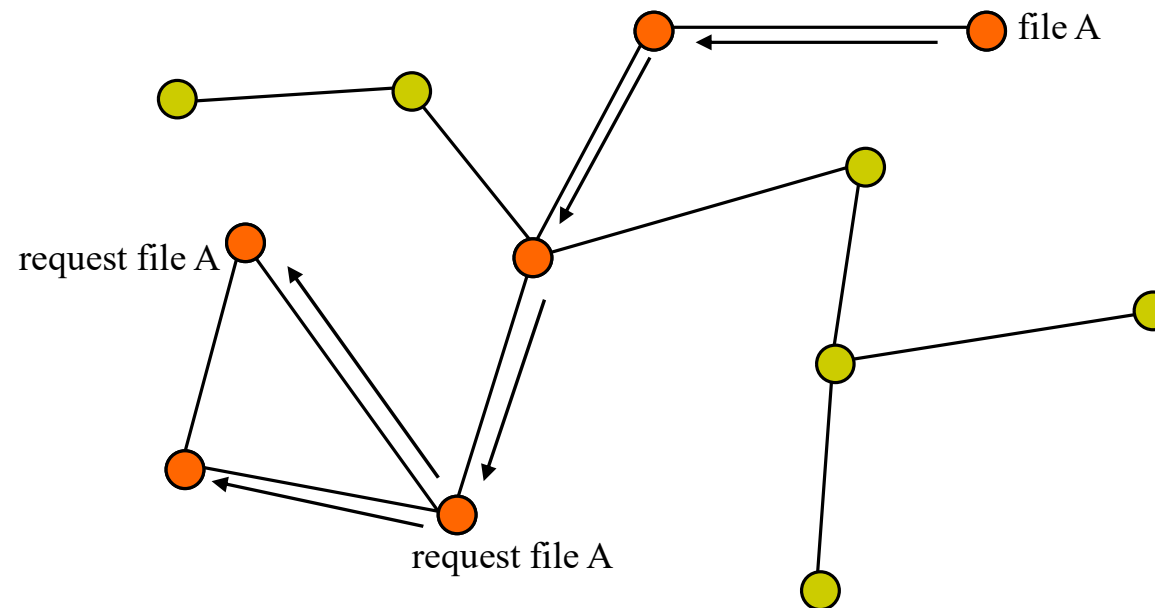
Traditional File Sharing Method: Client Server Architecture



Source is the bottleneck!

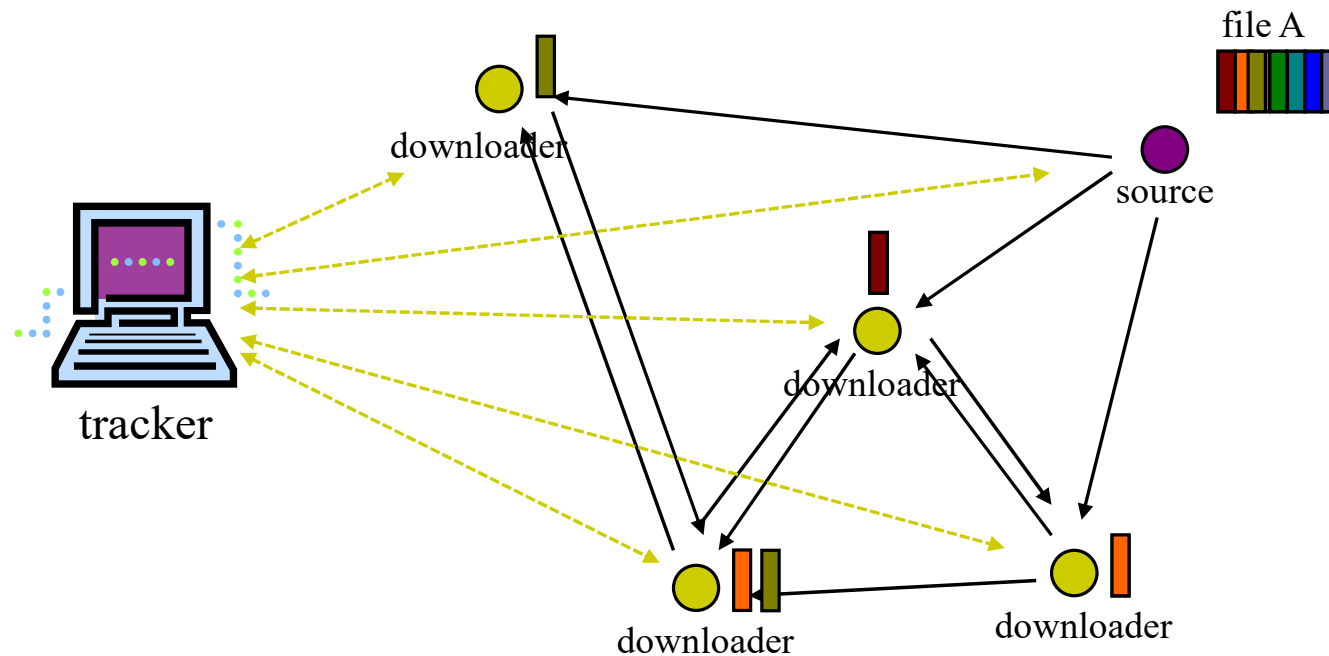
Bottleneck Exists Even in P2P

- ❑ The P2P architecture we have discussed so far helps distribute files; but, still, request is kind of client-server style
- ❑ A client can help distribute the content only when it has completed the downloading



BitTorrent

A peer helps others to distribute the file (by uploading the part of the file it has downloaded) while it is downloading.



A new peer needs to know who (downloaders) are in the swarm. This information is maintained at the tracker.

Terminology in BT

- ❑ **Torrent files** [*.torrent] are static metadata files containing information such as file size, name, checksum (using SHA1), and the IP of a **tracker**.
- ❑ **Downloaders (leechers)**: Peers that have only a part (or none) of the file.
- ❑ **Seeds**: Peers that have the complete file, and choose to stay in the system to allow other peers to download
- ❑ **Tracker**: peer that coordinates file distribution.
 - Alternatively, in a trackerless system (decentralized tracking) every peer acts as a tracker.
 - Implemented by the BitTorrent, µTorrent, BitComet, KTorrent and Deluge clients through the distributed hash table (DHT) method.

Piece

- ❑ A file is divided into pieces of fixed size, typically 64KB-4MB.
- ❑ Each downloader reports to all of its peers what pieces it has.
- ❑ To verify data, Hash codes are used for all the pieces, included in .torrent files
- ❑ Selecting pieces to download in a good order is very important for good performance.
 - **Rarest First**: Peers generally download pieces with the fewest number of owners.
 - Slow transfer rate from some peers may delay a download's finish.
 - **Endgame Mode**: If a download is close to its end, it will send request to all peers for the missing pieces.

Choking

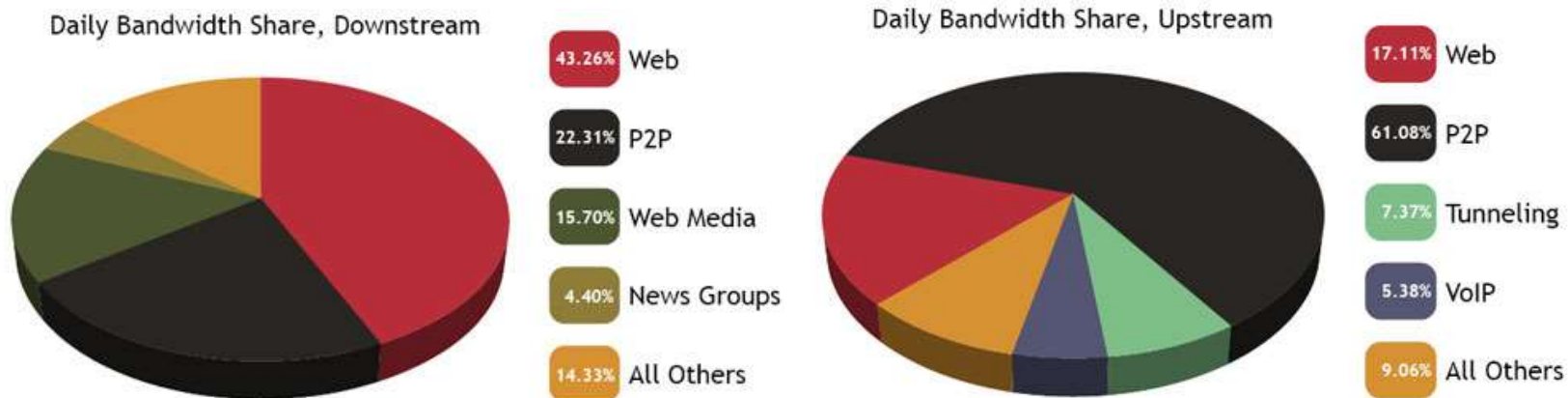
- ❑ “Choking” is to temporary refuse to upload to someone (but download can continue)
- ❑ A BT peer always unchokes some number of peers (default : 4), so the issue is which peer to unchoke.
- ❑ BT’s choking algorithm uses “*tit-for-tat*” strategy: the more you upload, the more you download
 - Free riders will be “choked”
 - Many more interesting strategies to study

Optimistic Unchoking

- ❑ Each peer has a single '*optimistic unchoke*', which is the peer that is uploaded regardless of the current download rate from it. This peer rotates every 30s
- ❑ Reason:
 - To discover if some currently unused connection is better than the ones being used

No. 1 Bandwidth Consumption in old days

- ❑ According to [CacheLogic](#), BT accounted for 35% of all traffic on the Internet in 2004.
- ❑ As of 2009, P2P traffic ranges from 43-70% depending on the source and region ([ITWorld 2009.02.18](#))
 - Below is one of the reports (by [Sandvine](#), 2008):



Legal Issues and Privacy

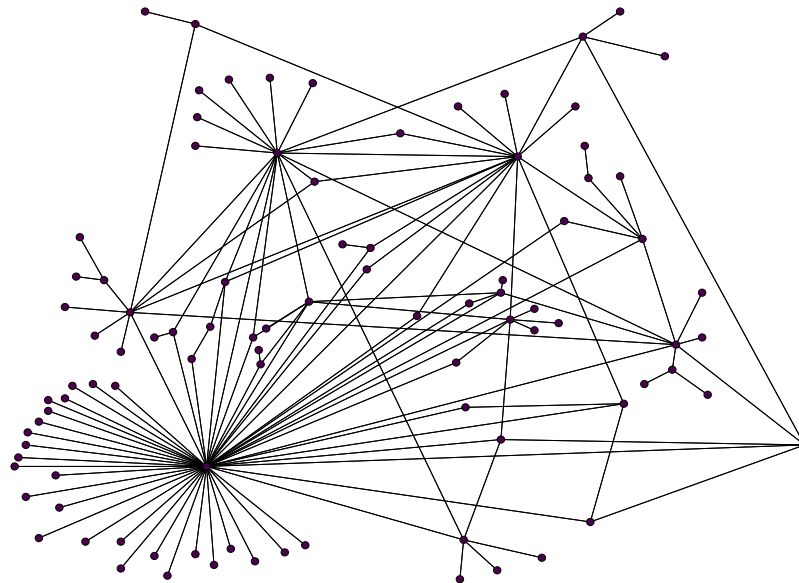
- ❑ BT itself does not offer a search facility to find files by name.
 - The torrent files contain the meta data information, and are distributed via a third channel.
- ❑ BT makes no attempt to conceal the host ultimately responsible for facilitating the sharing, tracking, or downloading.
 - Yet, seeds and trackers can be located in a jurisdiction where copyright is not useful.
 - With the distributed hash table (DHT) technology, trackers are no longer required.

Part II: Structured P2P Networks & DHTs

- ☐ Chord
- ☐ CAN
- ☐ Tapestry
- ☐ Skip Graph
- ☐ Kademlia
- ☐ Koorde

Unstructured P2P Networks

- ❑ Network's structure is arbitrary and so can not provide usual information for guiding query path
- ❑ Finding an object is kind of conducting a “blind search,” albeit there are some heuristics to improve efficiency



Structured Overlay

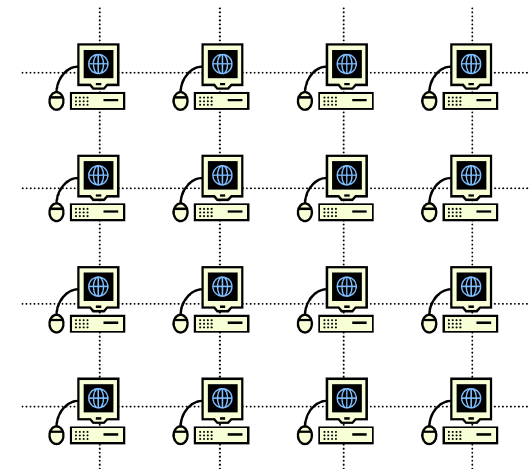
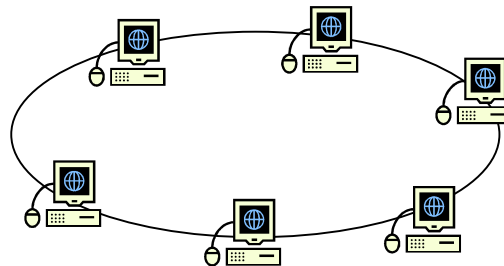
❑ Structured Overlays:



- Overlays are structured in some well-formed topology
 - ring, grid, tree, hypercube
- The structure then provides some knowledge for inferring direction of routing.

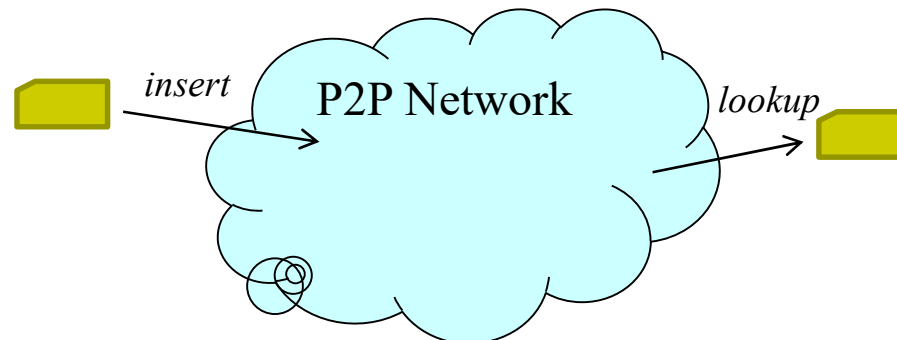
❑ Challenges:

- How to map objects to the nodes?
- How to arrange peers into the structure?
- How to maintain the structure?
- How to cope with dynamics?



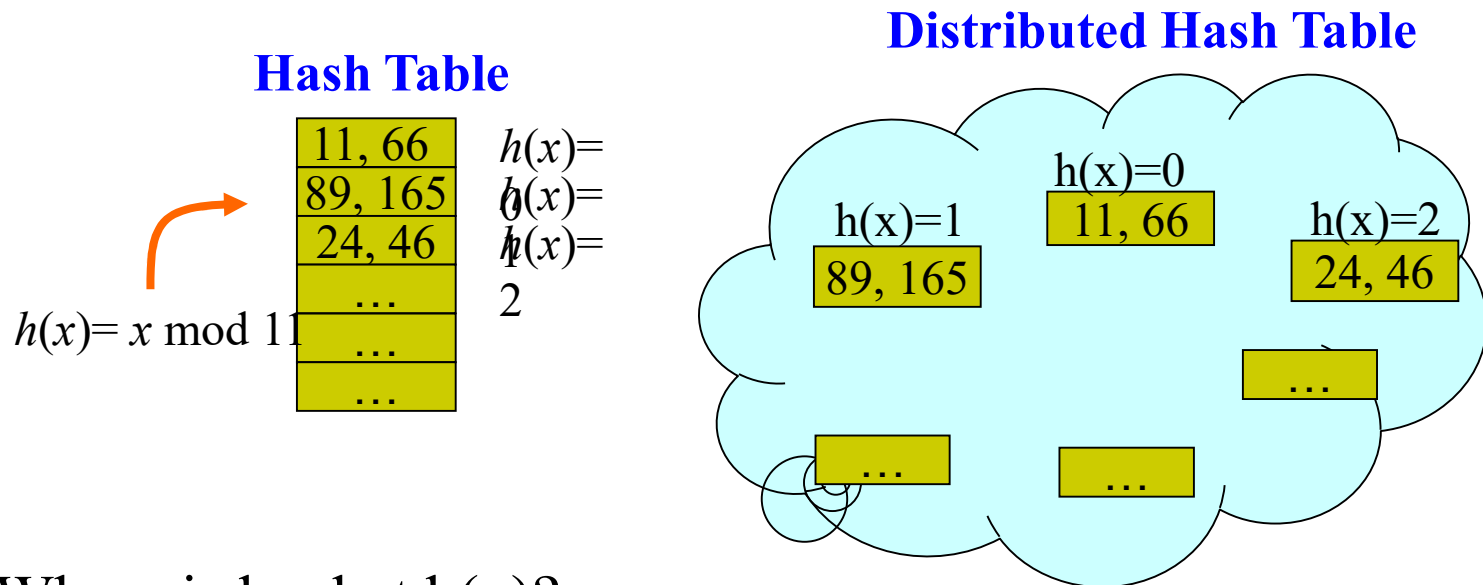
Lookup Protocol

- ❑ Fundamental component in structured P2P overlays for retrieving information/objects.
- ❑ Two primitive operations to manage key-value pairs
 - *insert(key, value)*
 - *value = lookup(key)*
 - key can be a hashed value of a file name
 - value can be a record holding metadata of the file
- ❑ The lookup protocol can be used by a file system to manage storing, replicating, caching, retrieving, and authenticating of files.



routing in the P2P network depends on the topology design

Distributed Hash Tables



❑ Where is bucket $h(x)$?

■ In centralized system

➤ trivial (e.g., an array of pointers to buckets, $A[h(x)]$)

■ In distributed system



➤ require nodes to maintain routing tables so as to cooperatively answer queries that may arise from anywhere in the network

➤ **query** becomes a **routing problem** if the object host id is known

➤ a **structured topology** helps a node to determine routing direction

Chord:

A Scalable Peer-to-peer Lookup Protocol for Internet Applications

Reading: Chord: a scalable peer-to-peer lookup protocol for internet applications. Ion Stoica. et al., IEEE/ACM Transactions on Networking, Vol. 11, Issue 1, Feb. 2003

Chord: a scalable peer-to-peer lookup protocol for internet applications

[I Stoica](#), [R Morris](#), [D Liben-Nowell](#)... - IEEE/ACM ..., 2003 - [ieeexplore.ieee.org](#)

... This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord

... Data location can be easily implemented on top of Chord by associating a key with each ...

☆ 儲存 ㊄ 引用 被引用 4361 次 相關文章 全部共 53 個版本

Chord: A scalable peer-to-peer lookup service for internet applications

[I Stoica](#), [R Morris](#), [D Karger](#), [MF Kaashoek](#)... - ACM SIGCOMM ..., 2001 - [dl.acm.org](#)

... This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord

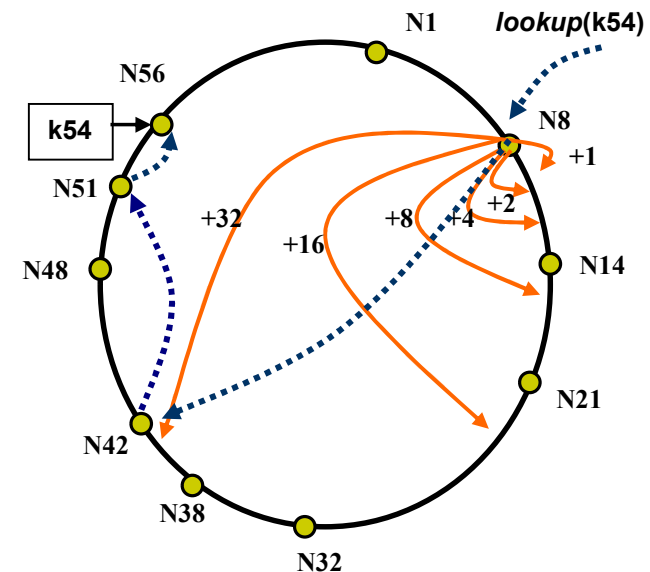
... Data location can be easily implemented on top of Chord by associating a key with each ...

☆ 儲存 ㊄ 引用 被引用 15184 次 相關文章 全部共 328 個版本 ㊄

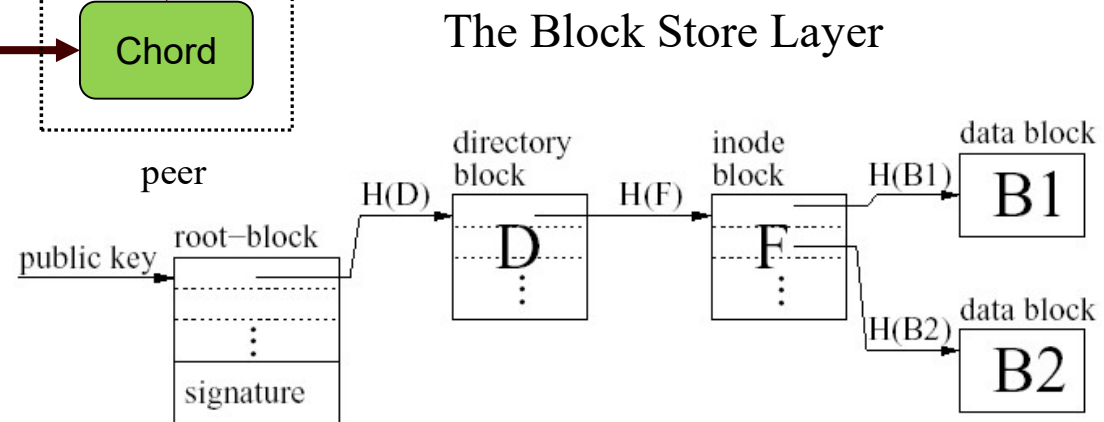
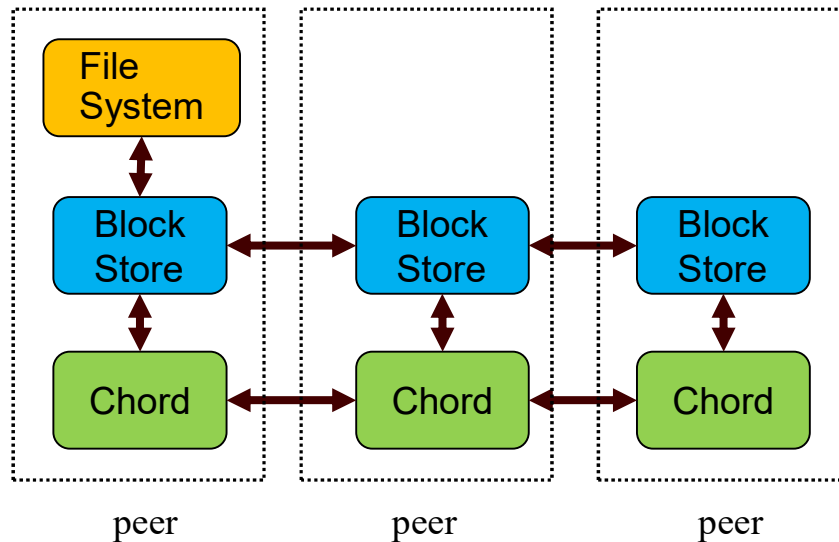
(as of 2023.03.16)

Chord Lookup Service

- ❑ Support for just one simple operation: maps a key onto a node.
- ❑ Applications can be easily implemented on top of Chord.
 - Cooperative File System
 - DNS
- ❑ Efficient and scalable:
 - $O(\log N)$ messages per lookup
 - $O(\log N)$ state per node
 - N is the total number of servers
- ❑ Robust: survives massive changes in membership



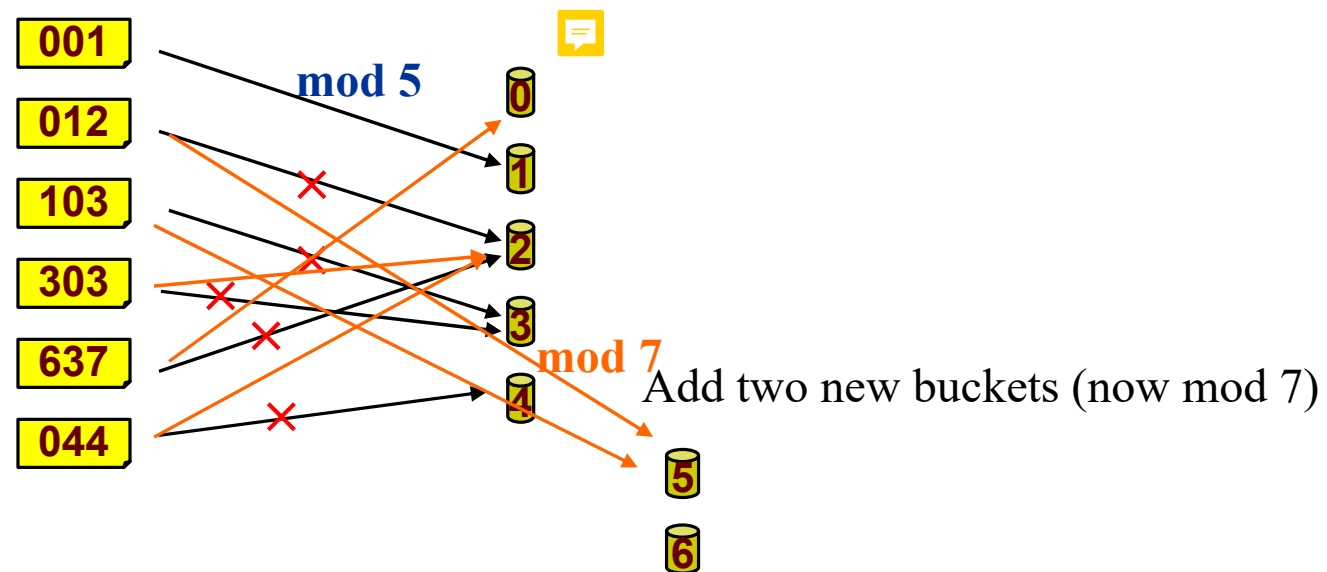
A Cooperative File System



- ❑ The root-block is identified by a public-key and signed by corresponding private key
- ❑ Other blocks are identified by cryptographic hashes of their contents

Hashing in Distributed Systems

- ❑ Generally used to distribute objects evenly into a set of servers, e.g.,
 - liner congruential function $h(x) = ax + b \pmod{p}$
 - SHA-1
- ❑ When the number of servers changes (p in the above case), almost every item will be hashed to a new location
 - Cached objects become useless in each server when a server is removed or introduced to the system.

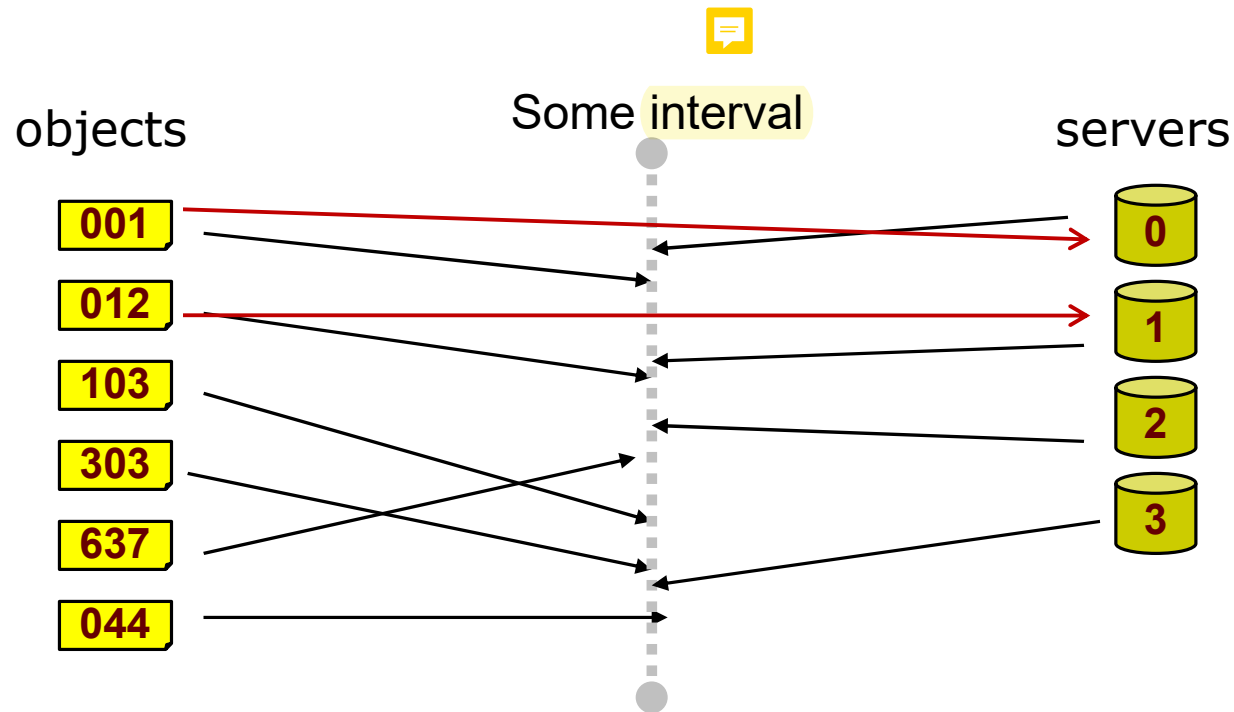


Consistent Hashing

- ❑ Load is balanced
- ❑ Relocation is minimum
 - When an N th server joins/leaves the system, with high probability only an $O(1/N)$ fractions of the data objects need to be relocated

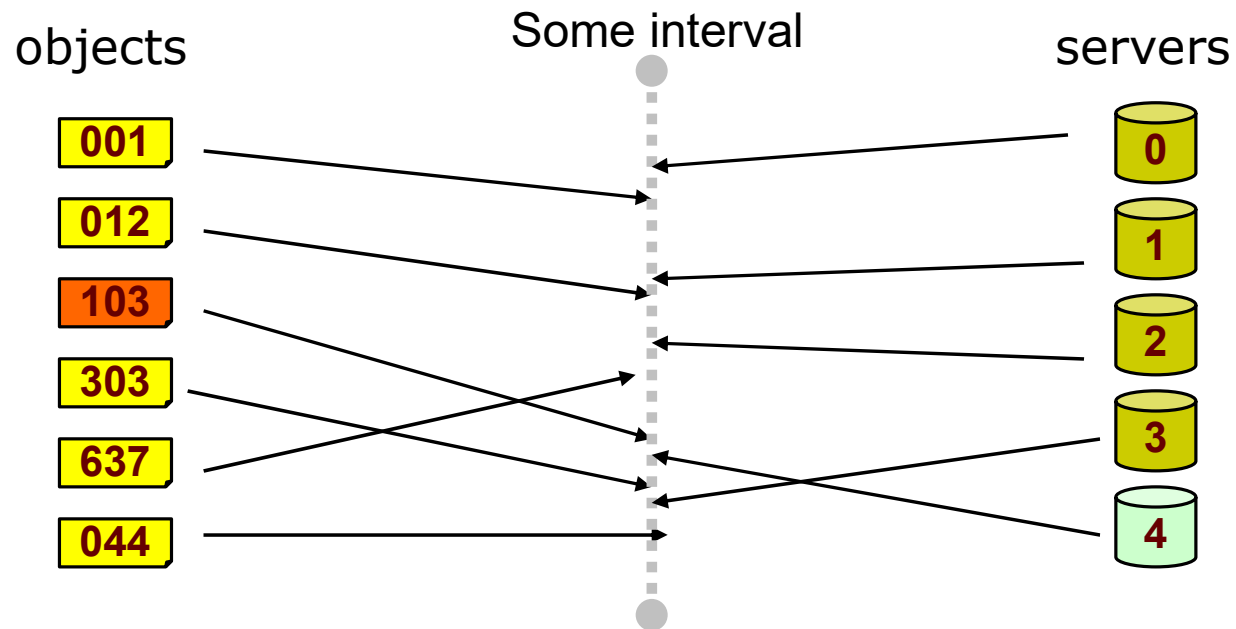


A Possible Implementation



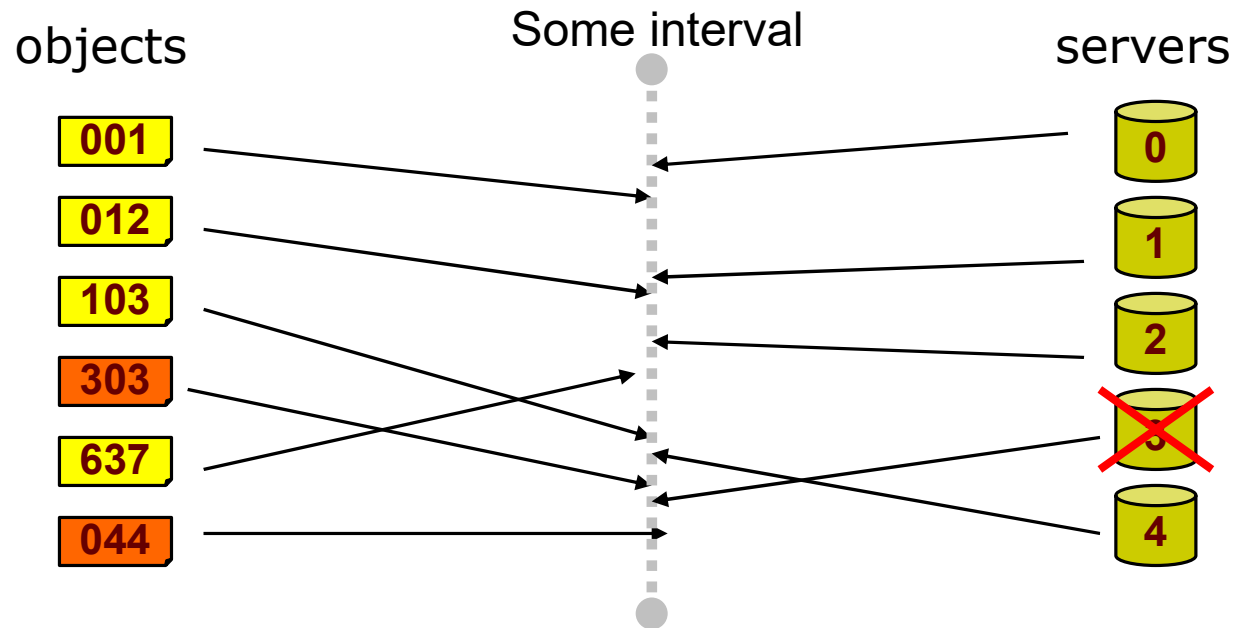
- ❑ Objects and servers are first mapped (hashed) to points in the same interval
- ❑ Objects are placed into the servers that are closest to them w.r.t. the mapped points in the interval.
 - e.g., $001 \rightarrow S_0$, $012 \rightarrow S_1$, $303 \rightarrow S_3$

When Server 4 joins



Only 103 needs to be moved from S3 to S4. The rest remains unchanged.

When server 3 leaves



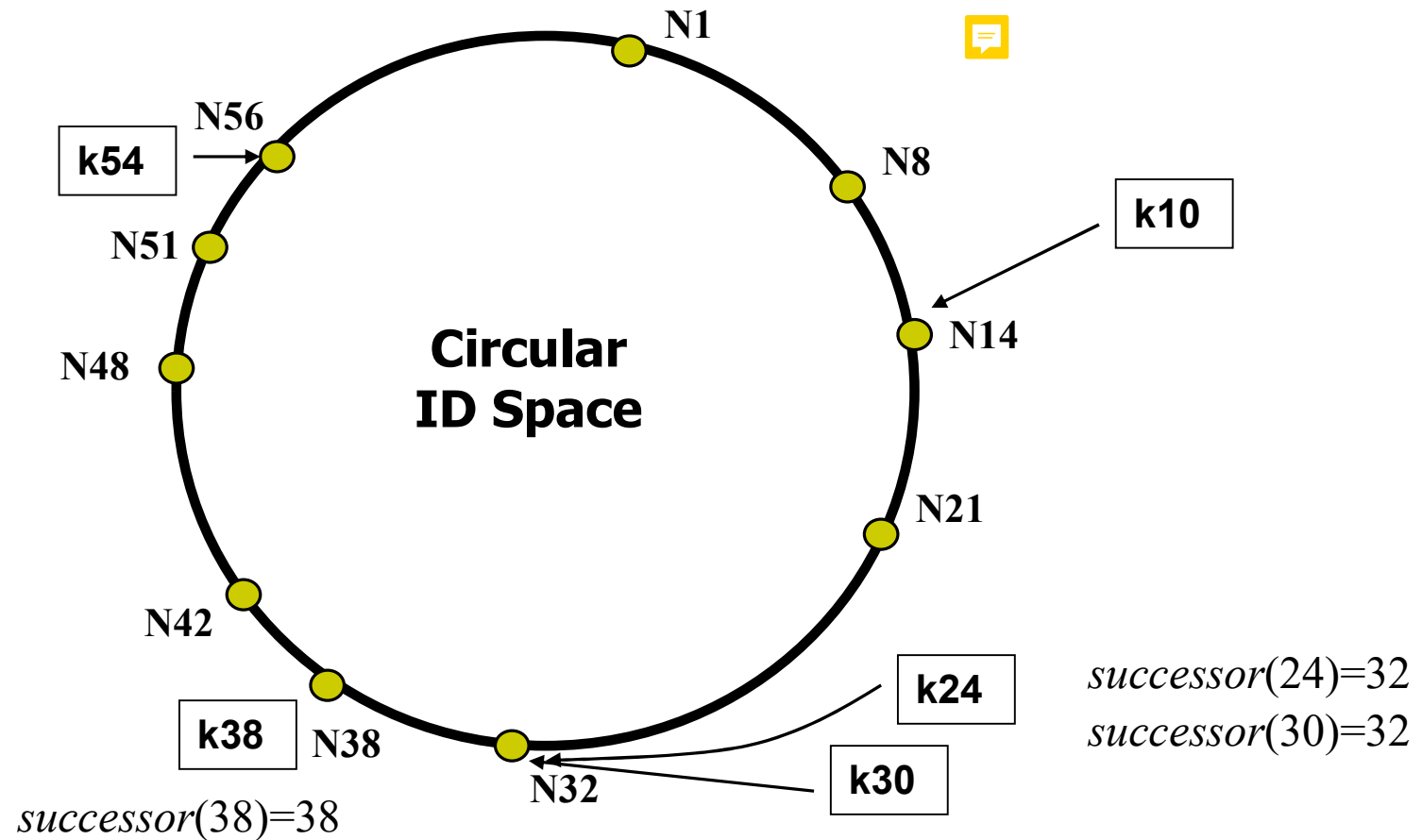
- ❑ Only 313 and 044 need to be moved from S3 to S4.
- ❑ In general, hashing allows each server to share about $1/N$ of the objects
- ❑ When a server is changed, about $1/N$ of the objects will be relocated

Consistent Hashing in Chord

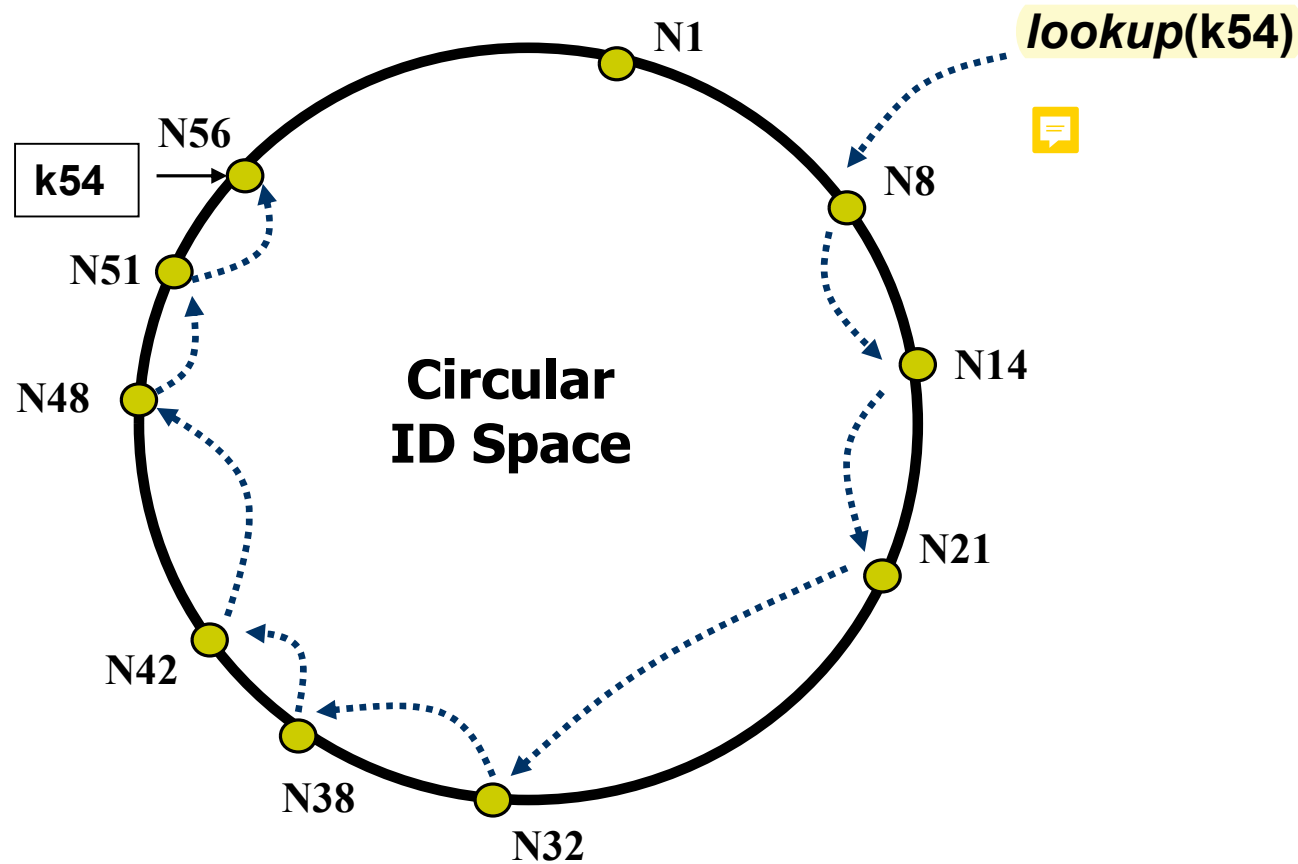
- ❑ Node's ID = SHA-1 (IP address (+port))
- ❑ Key's ID = SHA-1 (object's key/name)
- ❑ Chord views the IDs as
 - uniformly distributed
 - occupying a circular identifier space
- ❑ Keys are placed at the node whose IDs are the closest to the (IDs of the) keys in the clockwise direction.
 - *successor(k)*: the first node clockwise from *k* (including *k*).
 - Place object *k* to *successor(k)*.

MIT Chord

An ID Ring of length 2^6-1

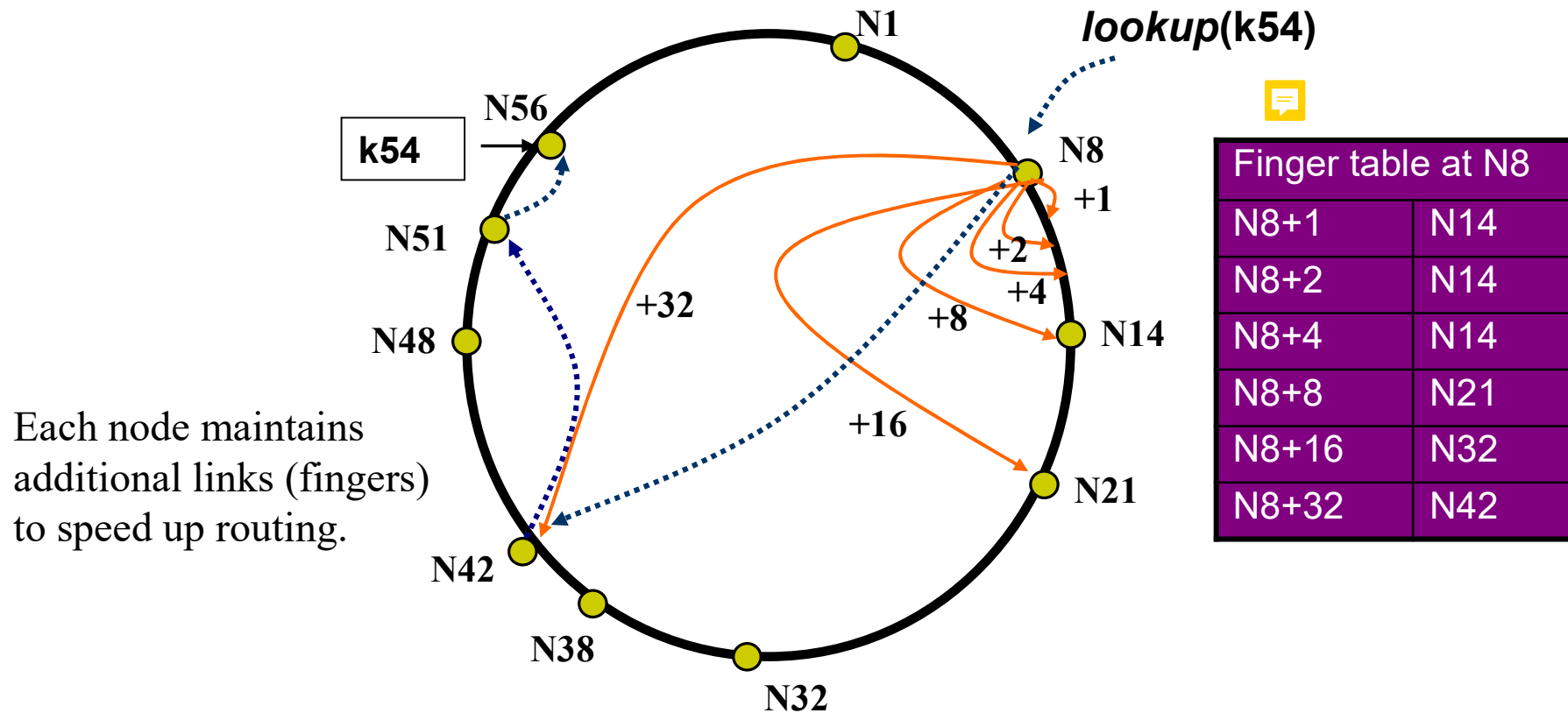


Simple Lookup



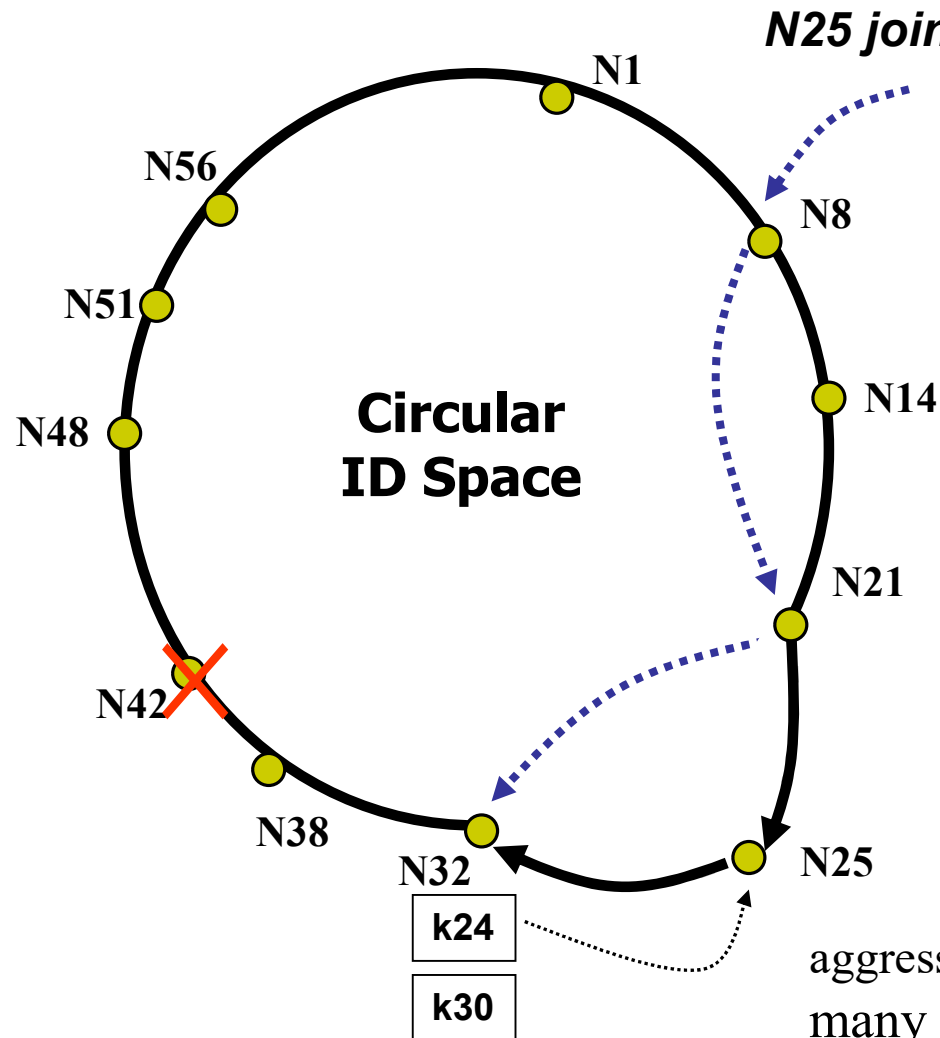
- ❑ Lookup succeeds if successors are correct
- ❑ Average of $n/2$ message exchanges

Scalable Lookup



- ❑ Routing is to approach the destination as much as possible without “over shooting”
- ❑ Each node can forward a query at least halfway along the remaining distance between the node and the target identifier.
- ❑ Lookup takes $O(\log N)$ steps.

Dynamics: Node Join/Exit



Finger table	
N8+1	N14
N8+2	N14
N8+4	N14
N8+8	N21
N8+16	N32
N8+32	N42

```
// join a Chord ring containing node n'
n.join(n')
predecessor := nil;
successor := n'.find_successor(n);
```



aggressive mechanisms requires too many messages and updates

Node Fails

- ❑ Can be handled simply as the invert of node joins; i.e., by running stabilization algorithm.

```
// periodically verify n's successor s and inform s of n
```

```
n.stabilization()
```

```
  check_predecessor();
```

```
  x := successor.predecessor;
```

```
    if ( $x \in (n, \text{successor})$ ) // successor has changed due to new join
```

```
      successor := x;
```

```
  successor.notify(n);
```

```
// n notifies s to be s' predecessor
```

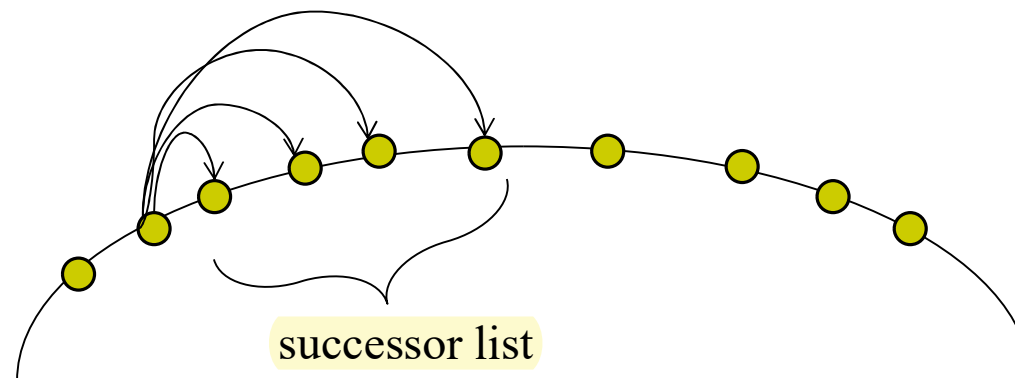
```
s.notify(n)
```

```
  if ( $\text{predecessor} = \text{nil}$  or  $n \in (\text{predecessor}, s)$ )
```

```
    predecessor := n;
```

Handling Failures

- ❑ Use successor list
 - Each node knows r immediate successors
 - After failure, will know first live successor
 - Correct successors guarantee correct lookups
- ❑ Guarantee is with some probability
 - Can choose r to make probability of lookup failure arbitrarily small



how to avoid putting too much storage load to a node's neighbor when the node leaves the network? “**virtual**” nodes

Chord Summary

❑ Advantages

- Lookup guaranteed to be completed in $O(\log N)$ steps
- Routing table size is only $O(\log N)$
- Robust---can handles large number of concurrent join and leaves

❑ Weakness

- Performance: routing in the overlay network can be more expensive than in the underlying network
 - No correlation between node Ids and their locality; a query can repeatedly jump from Taiwan to America, though both the initiator and the node that store the item are in Taiwan!
- Require some efforts to maintain the overlay

Building Systems on Top of Chord

- ❑ Note that Chord (or DHTs in general) can act either as actual data stores or merely as directory services storing pointers.

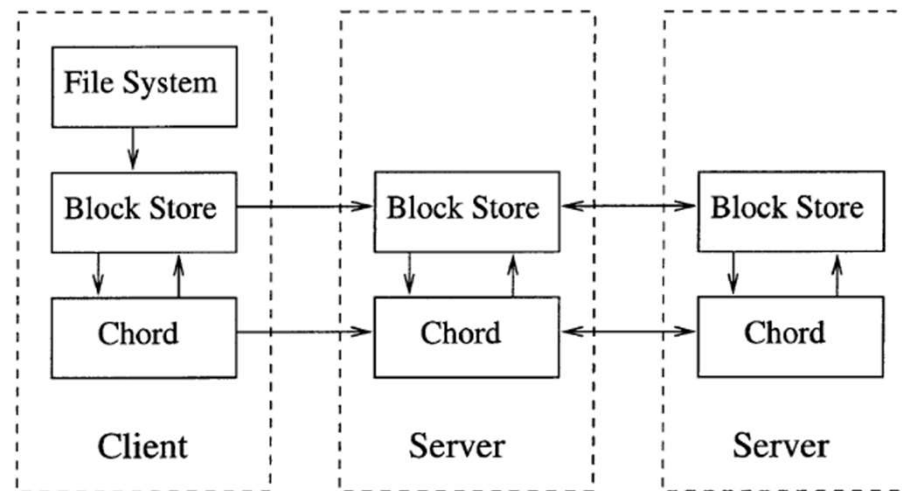


Fig. 1. Structure of an example Chord-based distributed storage system.

source: [Chord: a scalable peer-to-peer lookup protocol for internet applications](#). Ion Stoica. et al., IEEE/ACM Transactions on Networking, Vol. 11, Issue 1, Feb. 2003