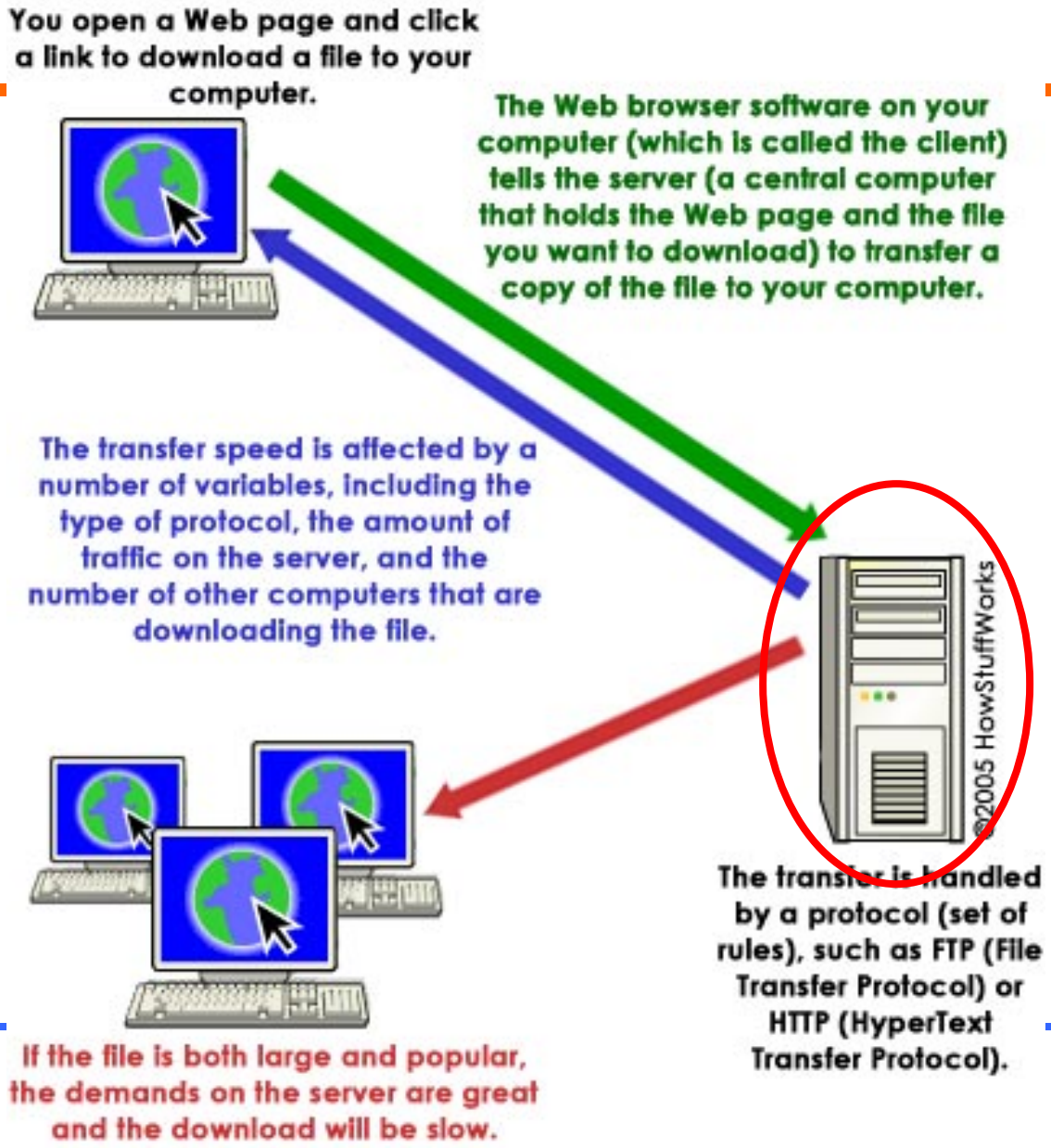
Peer-to-Peer Systems (P2P)

Traditional Client-Server Architect



What is Peer-to-Peer?

- P2P is a type of network that all computers have the equal capabilities and responsibilities (different from the client/server model)
  - The correct functioning of a system does not depend on availability of any central node
- P2P design ensures that nodes (peers) contribute resources to the system, as well as consume resources
- A key issue of P2P systems is the design of algorithms to place data across many nodes and subsequently retrieve data with high efficiency and availability

Two Types of P2P

- Purely distributed P2P
  - Control is purely decentralized
  - Peers act as both clients and servers
  - Peers have equal role/capability
  - Peer-peer communication is symmetric

- Hybrid
  - There exist central servers/super-peers for indexing the resources or allocating tasks in the system, but data are distributed on peers

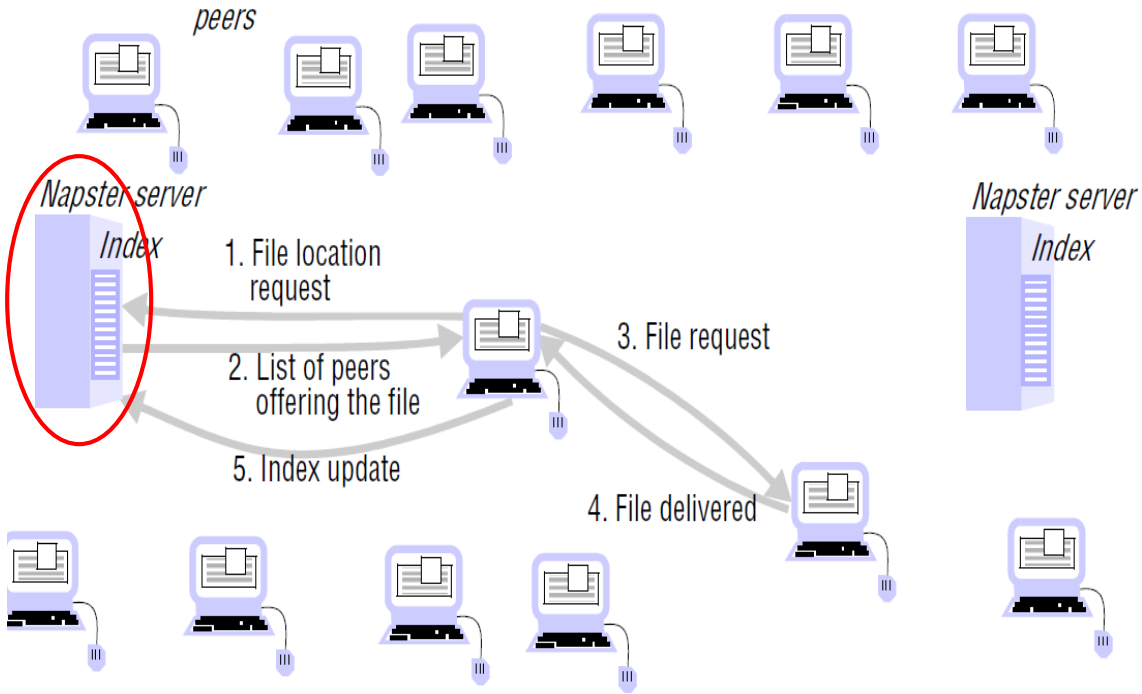
Generations of P2P

- 1st generation: Centralized indexes
  - Napster music file sharing
- 2nd generation: File sharing applications offering greater scalability, anonymity and fault tolerance
  - Freenet, Gnutella, BitTorrent
- 3rd generation: A middleware layer for application-independent management of distributed resources in a global scale
  - Pastry, Tapestry
  - bitcoin

Napster: A Music file sharing system

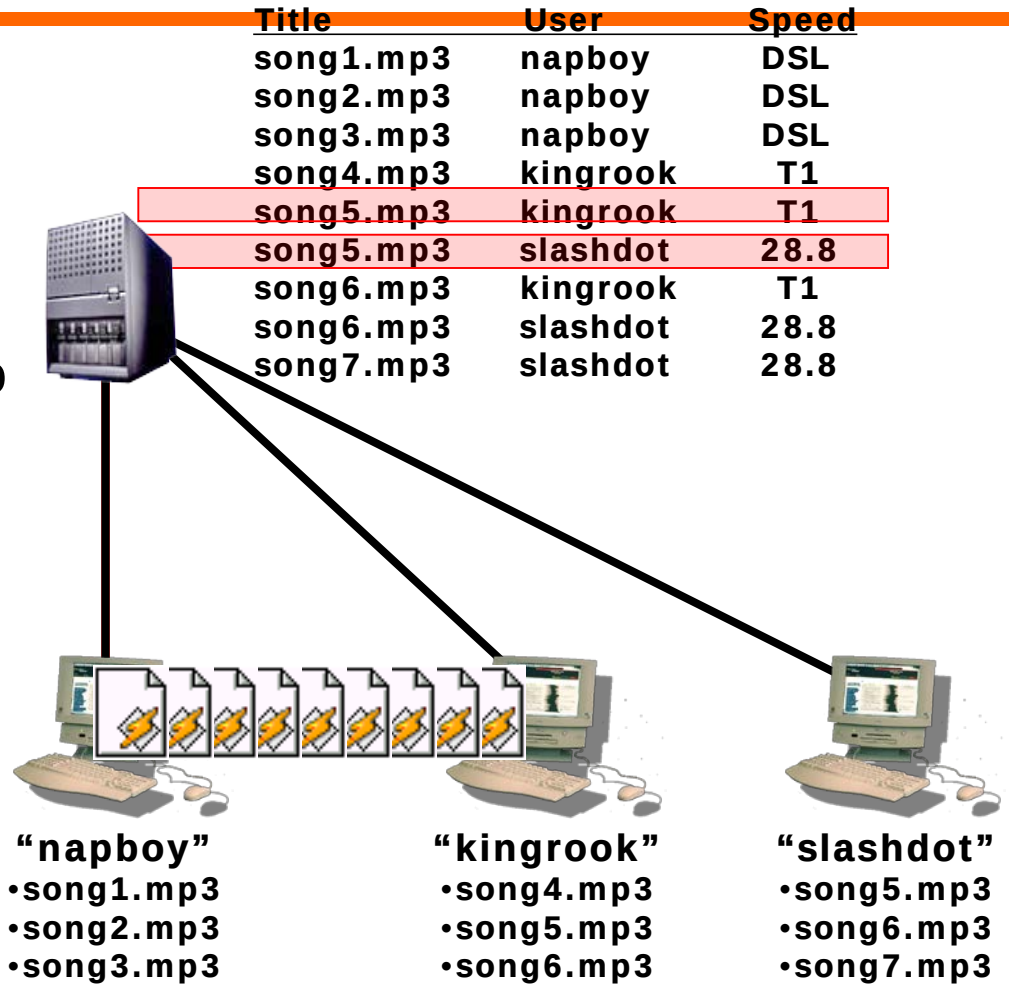
Napster: Files are stored in peers but indexing is done by a central svr

1. User requests the server by input file name in the client search-box
2. Server returns a list of links containing the file
3. User clicks a link to request download of file from a peer
4. The peer transfer the file to the user
5. User requests the server to add its files to the shared pool



Napster

- Napster uses centralized indexing, limited scalability of index search
- Napster was shutdown due to legal issue of hosting copyrighted materials
- Anonymity of content receivers and providers becomes a concern of P2P design



Lessons from Napster

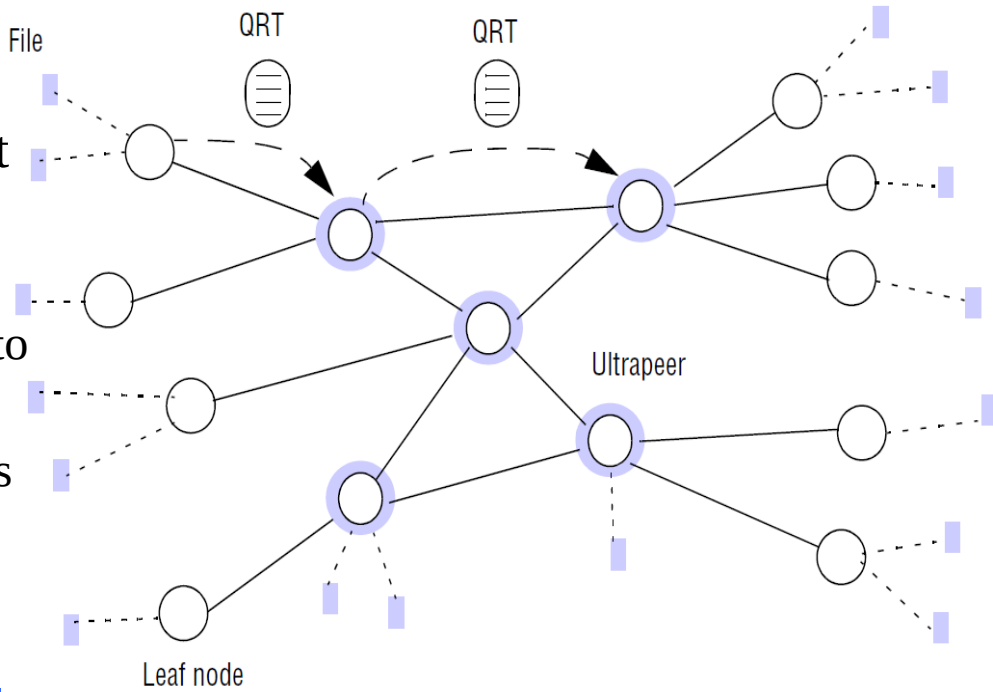
- Napster's design is only suitable for music service:
  - No update to data files
  - Temporary unavailability is tolerable
- Is it suitable for other file sharing applications?

Gnutella: a file sharing system

- Initial version of Gnutella uses a pure P2P routing method
  - Each node keeps active connection with a number of other peers (around 5)
  - For search, a node broadcasts a request to all its actively connected peers, which further forward the request to others until TTL expires (usually 7)
- Gnutella V0.6 introduced ultra-peers to reduce routing complexity
  - Some peers with additional resources are promoted to ultra-peers
- Each ultra-peer connects to at least 32 ultra-peers (heavily interconnected), limiting max query-hops to 4
  - Ultra-peers form a core-network to route user's queries
- The rest of peers are leaf nodes, each connecting to a small number of ultra-peers (typically 3)
  - Leaf nodes are not responsible for routing for others

How Gnutella Works?

- Leaf nodes and ultrapeers use a Query Routing Table (QRT), a table for matching hashed keywords to file locations
  - Each leaf produces a QRT for its local files and sends it to its ultra-peers
  - A ultra-peer merges QRT of its leaves & its own, exchanges with its connected ultras
- A query is done by hashing the words of file name into numbers (e.g., "Gnutella P2P" → {09, 76}) and exact matching with the local QRT
  - If a matching is found in QRT, the node-addr of queried file is sent back to the user directly via UDP
  - Otherwise, the query is broadcast to its direct neighbor ultras
- The user can then negotiate the file transfer from the nodes having the file



Improvement of Gnutella

- Gnutella enhances anonymity of files and their owners by hashing file names into numbers
- The decentralized QRT of Gnutella makes any authority difficult to shutdown a particular file (content)
  - It's the users who decide which content they want to host
- To avoid flooding among ultrapeers, a node sends the query to one ultrapeer at a time, and waits for a result for a certain period of time before requesting another untrapeer
- Gnutella's routing is unreliable due to the fact peers or ultrapeers may frequently leave or join the system
  - Making the routing structure unstable

BitTorrent

- BitTorrent is a P2P protocol that enables fast downloading of large files (video files)
- BitTorrent split files into fixed-size (256K, 512K or 1M) chunks and makes chunks available at various sites across P2P network
- Clients can download chunks of a file simultaneously from multiple sites (peers) that already have them
  - The index of the file (specifying which nodes store chunks of the file) is stored in a node called tracker
  - There are many public trackers on the internet. Users can choose different trackers to use
  - The IP addr of the tracker for a file is specified in a meta-file with extension name '.*torrent*'
    - For example, a data file 'Panda.wmv', its metafile name is 'Panda.torrent'

Terminologies in BitTorrent

xxx.torrent file – A meta-data file about a data file (.torrent is the file extension name)

Tracker - A centralized server that manages (keeps track of) the download of a data file

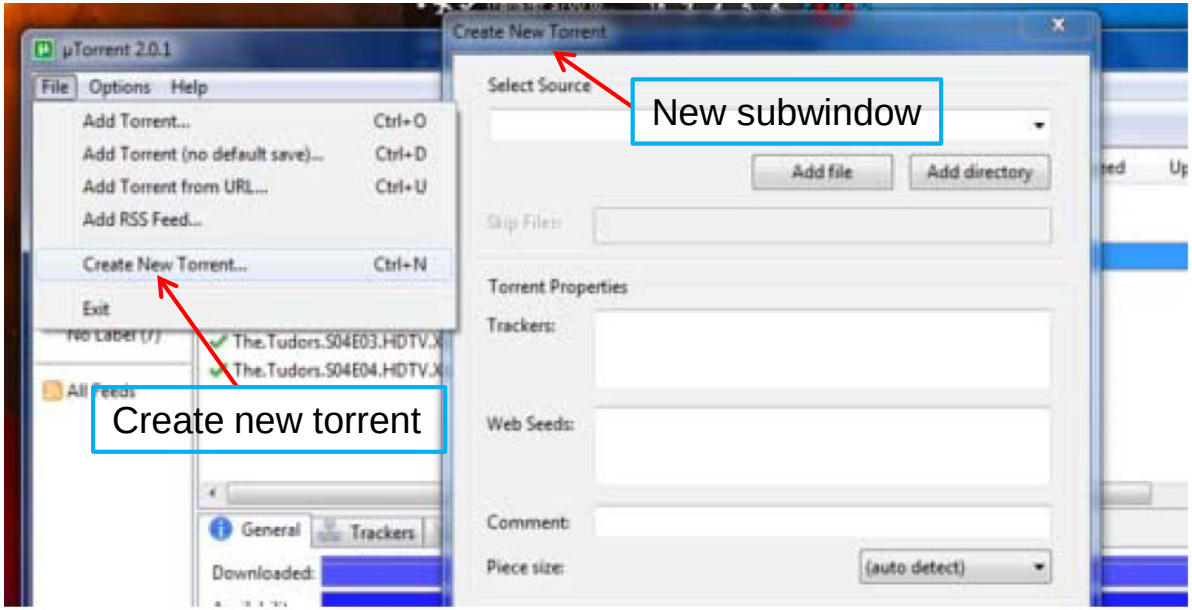
Leecher – A peer that downloads a file (holds part of the file)

Seeder - A peer that holds a complete copy of a file

Swarm - A group of peers involved with the download of a file, including the tracker, seeders and leechers

Upload a file in BitTorrent

- When a user (seeder) wishes to make a file (say *Panda*.wmv) available in BitTorrent, he creates a metadata file *Panda*.torrent in BitTorrent client
- The '.torrent' file contains information:
  - Name and length of the file, checksum of each chunk
  - Location of a tracker specified as URL
- The user can then email metafile *Panda*.torrent to friends or embed it in a web-link for others to download
- The .torrent file enables the download of data file in BitTorrent

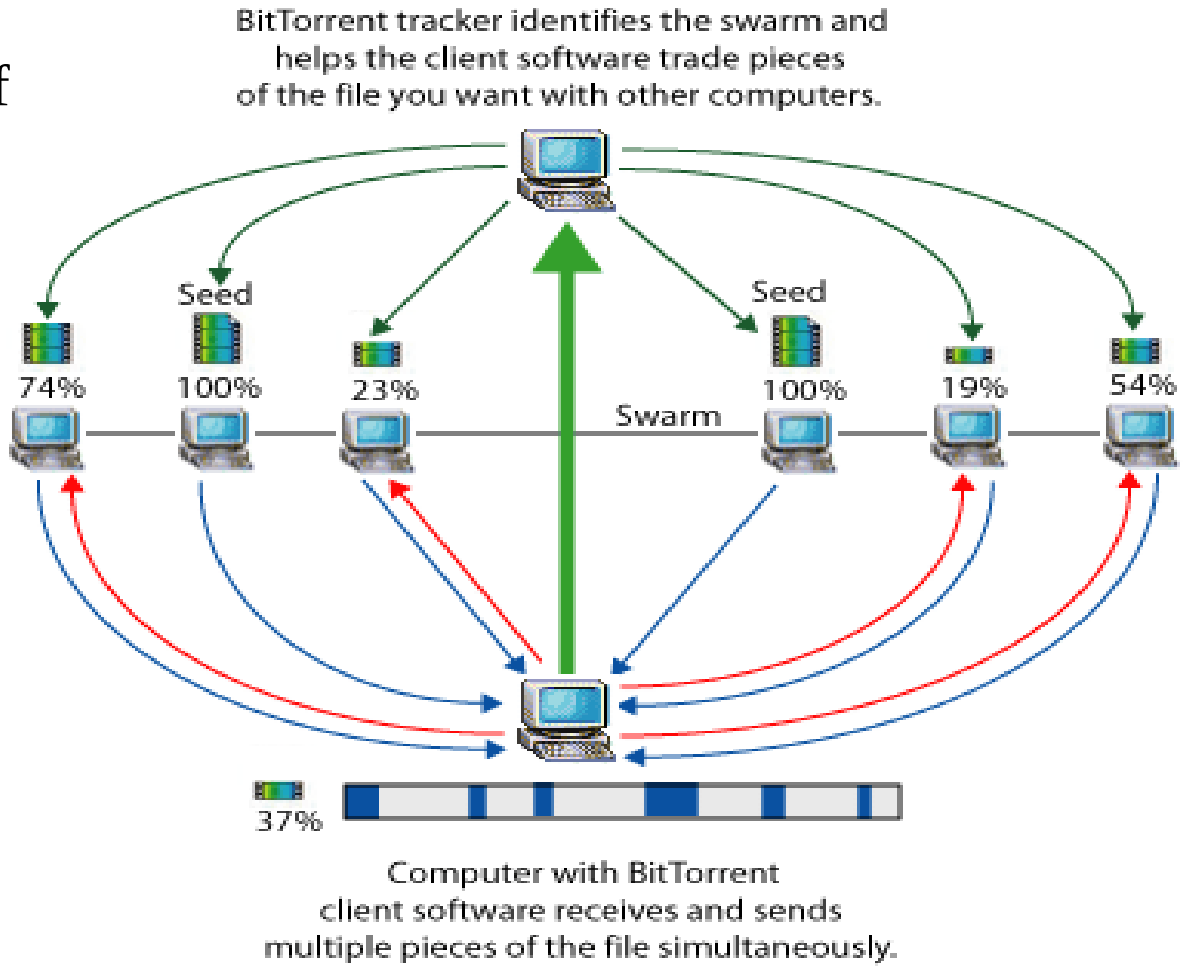


BitTorrent P2P Downloading

1) When user clicks a web-link embedded with a '.torrent' file, this meta-file is downloaded and opened by a BitTorrent-client (e.g., *uTorrent*, which should be already installed in the local site)
2) BitTorrent-client requests the tracker to send over information of seeders and swarm of the data file
3) BitTorrent-client starts to download chunks from multiple sites in swarm simultaneously, and in the meantime it makes its downloaded chunks available to others
   - The tracker keeps track of the downloading process of the client
   - Sharing starts immediately when a chunk is available (no need to wait for the whole file available)
   - Download speed get faster and faster as more users download the file

How tracker, swarm and client work

- The tracker keeps track of downloading of the client (client reports to tracker)
- There should be at least one seeder in a swarm
- The chunks downloaded to client are in parallel and out of order, but they eventually make up the whole file



Two Policies to Improve Performance in BitTorrent

Leeching problem refers to the clients who download without sharing. BitTorrent relies on clients to share their files with others.

- **Tit-for-Tat** is a policy that gives higher priority to the peers who previously or currently uploading files for others
  - BitTorrent client keeps track of your upload/download ratio
  - Others can receive xxx.torrent file and data chunks from your site after your download (so long as xxx.torrent file is still at your site)
  - You can remove xxx.torrent files to stop sharing. It's better do it after your upload/download ratio reaches 1
- **Rarest-First** is the policy that a peer schedules downloading the "rarest chunk" first among all the chunks
  - Fast populating the rarest chunks to make them available to others
  - Speed up the spread of rare chunks (unchoke the bottleneck peers)

Current status of BitTorrent

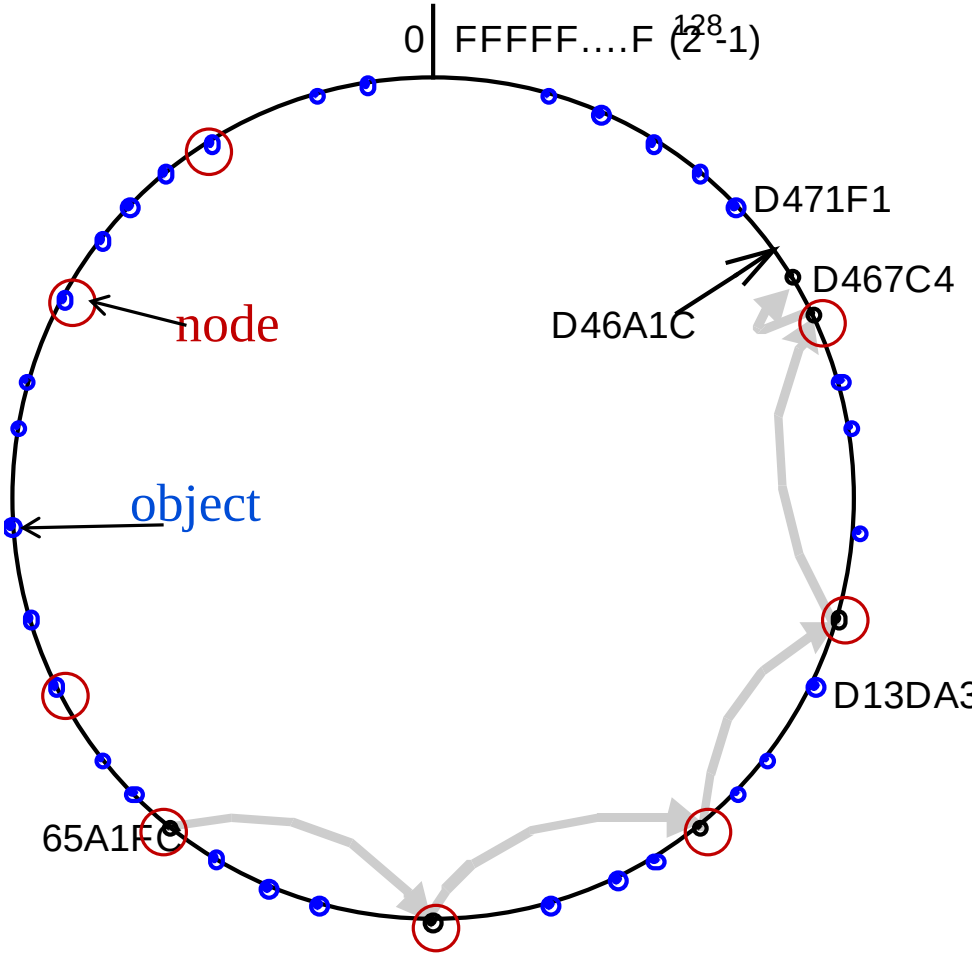
- BitTorrent is still the most popular protocol for transferring large video files, such as video files or TV shows
- BitTorrent had 15–27 million concurrent users at any time and was responsible for 3.35% of all worldwide bandwidth at 2012
- BitTorrent does not offer its users anonymity nor security
  - Users (participants) IP addresses can be obtained…

P2P Middleware

- Objects are identified by a globally unique ID (GUID) – usually derived from a secure hash function from the name or part of the object
  - Anonymity, self-certifying, low clashing probability, etc.
- A node, wishing to make an object available to P2P, computes the GUID of the object and announces it to the routing overlay
  - The object now becomes accessible to others
- The overlay routes client's requests to the node that holds the address of the object or the object itself
  - Fault tolerance of locating objects (redundancy of routing)
  - Bounded number of hops for locating objects
- Nodes may dynamically join and leave the P2P system
  - For a node joining, it joins the routing overlay
  - For a node leaving, its responsibility is redistributed to others

A linear routing scheme for Pastry

- **Both nodes and objects** are assigned 128bits GUIDs via a hash function and mapped to the same logical ring-space
- The GUID space is treated as circular: node 0 is adjacent to node (2^{128}-1)
- Each node stores a leaf set of size 2*l* (*l* = 4), containing pairs (GUID, IP-addr) of –*l* and +*l* neighbors from this node
- Routing selects the node in the leaf-set that is closest to *dest* as the next hop
  - e.g. a user at node 65A1FC wishes to locate object D46A1C
  - The address of object D46A1C is contained in the leaf set of node D467C4
- The routing scheme is not scalable
  - max (N/2)/*l* hops to locate an object

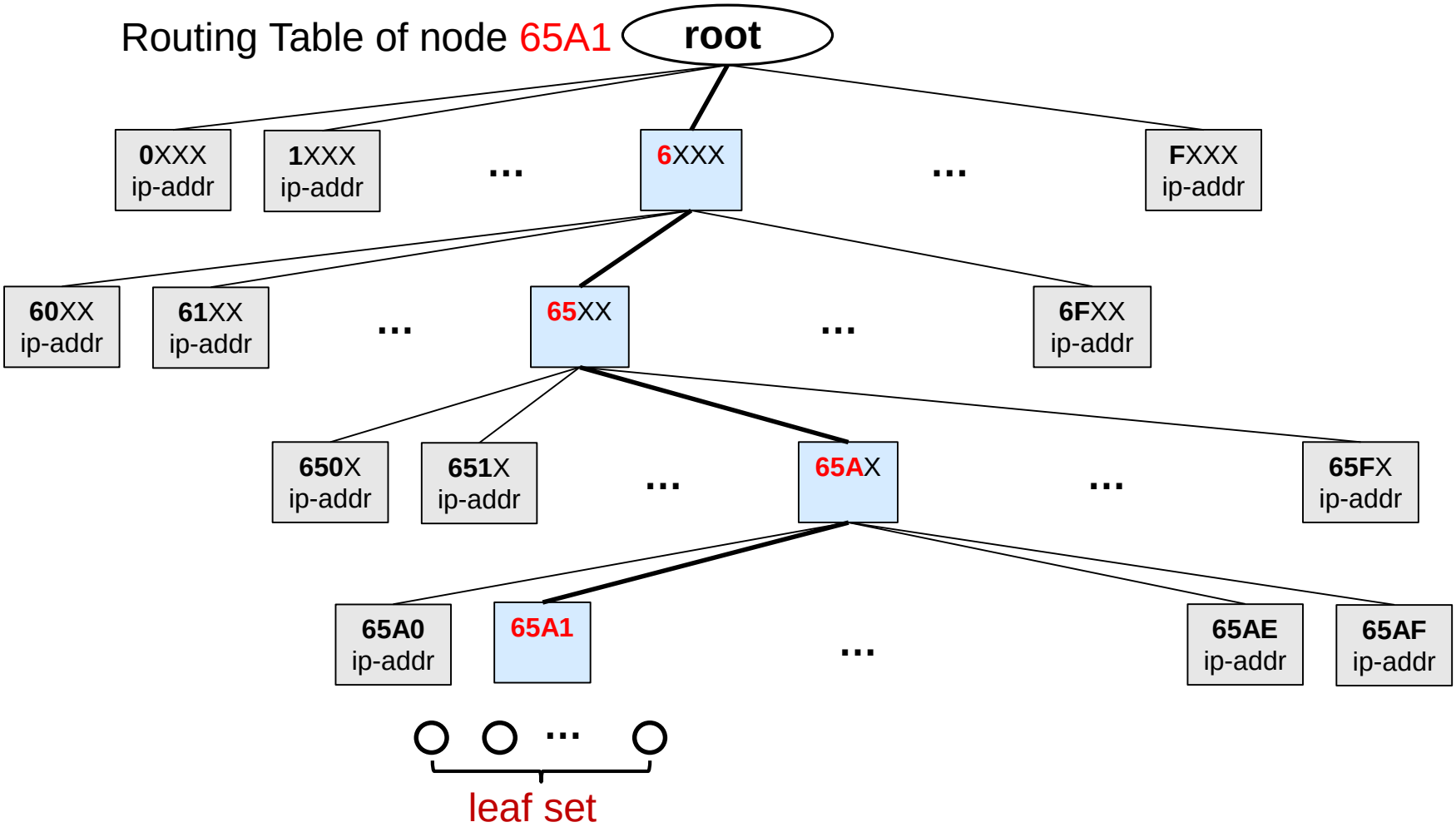


The tree routing scheme for Pastry

p =	Routing Table for node GUID 65A1	*GUID prefixes and corresponding nodehandles n*														
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	n	n	n	n	n	n		n	n	n	n	n	n	n	n	n
1	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6F	6E	6F
	n	n	n	n	n		n	n	n	n	n	n	n	n	n	n
2	650	651	652	653	654	655	656	657	658	659	65A	65B	65C	65D	65E	65F
	n	n	n	n	n	n	n	n	n	n		n	n	n	n	n
3	65A0	65A1	65A2	65A3	65A4	65A5	65A6	65A7	65A8	65A9	65AA	65AB	65AC	65AD	65AE	65AF
	n		n	n	n	n	n	n	n	n	n	n	n	n	n	n

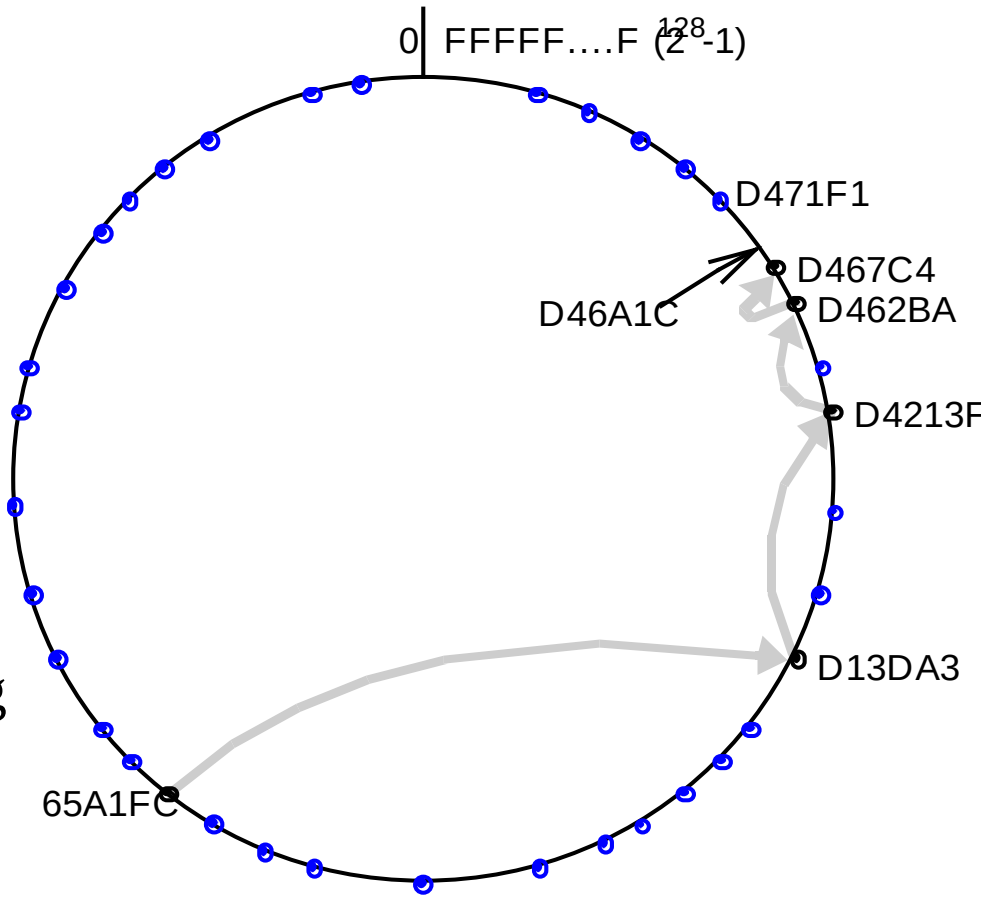
The routing table is located at a node whose GUID begins 65A1. Digits are in hexadecimal. Then s represent [GUID, IP address] pairs specifying the next hop to be taken by messages addressed to GUIDs that match each given prefix. Grey-shaded entries indicate that the prefix matches the current GUID up to the given value of p: the next row down or the leaf set should be examined to find a route. Although there are a maximum of ~~128~~ 32 rows in the table, only $\log_{16} N$ rows will be populated on average in a network with N active nodes.

The Tree of Pastry's Routing Table



Pastry Algorithm

- GUIDs are represented in Hex-numbers (32 Hex-bits)
  - Routing table of a node has 32 rows, each with 15 entries
- Routing is done by prefix-matching of object GUID with its own node GUID
  - Each hop matches part of the prefix and moves closer to the *dest*
- Each routing hop moves towards 1/16 of the existing nodes, leading to ~$\log_{16}$N hops to reach the *dest*



Pastry Routing Algorithm

To route msg M to node D, where $R[p,i]$ is row p, col i of A's routing table:

// D is the destination, and A the local node

If ($L_{-l} < D < L_l$) { // D is in the leaf set L or ($D == A$)

Forward M to L_i in the leaf set whose GUID closest to D or node A.

} *else* { // route M to a node with a closer GUID

Find $p \equiv$ the max length of common prefix of D and A,

and $i \equiv$ the $(p+1)^{th}$ Hex-digit of D

If ($R[p,i] \neq null$)

Forward M to $R[p,i]$

else { // no entry in the routing table, rare case!

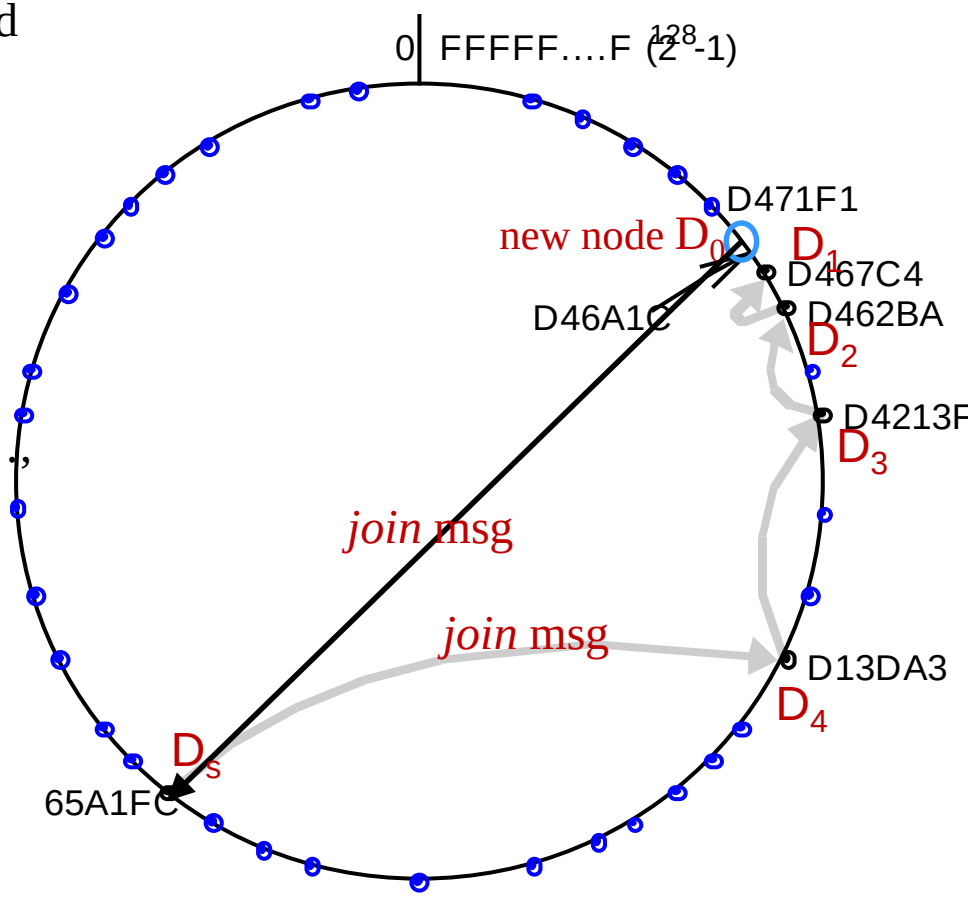
Forward M to any node in L or an entry in R with max common prefix and whose GUID is numerically closer to D than A

}

}

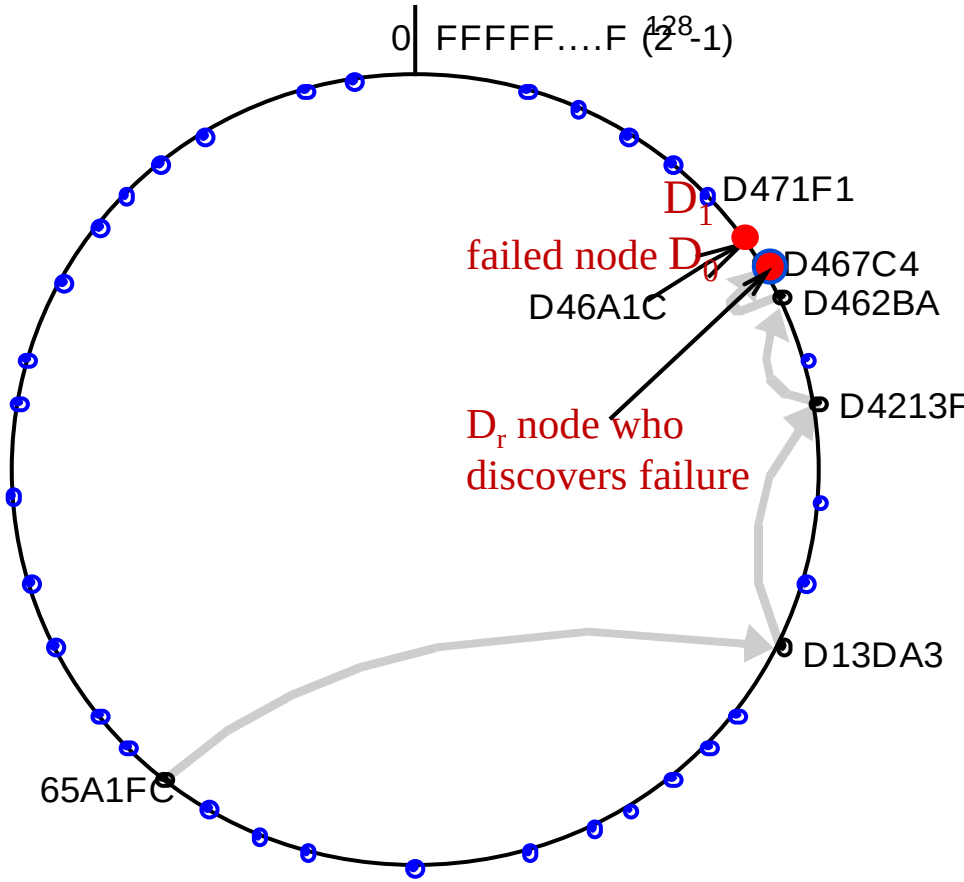
Node Joining Pastry: Routing Table Construction

1) New node D_0 computes a GUID for itself and finds a nearby Pastry node D_s with min network delay/hop (in Internet distance)
2) D_0 sends a *join* msg to D_s, and D_s dispatches the *join* msg via Pastry as **search for** D_0. Pastry routes this *join* msg eventually to a node D_1 who is closest to D_0 in GUID space
3) Nodes that the *join* msg traverses, D_s, D_{s-1}, …, D_1, all sends their relevant parts of tables to D_0, from which D_0 constructs its own table
4) Since D_s is nearest to D_0 in Internet, D_0 sets its 1st row of table to D_s's 1st row, 2nd row to D_{s-1}'s 2nd row, …; and its leaf set to D_1's set
5) D_0 sends its own GUID and IP-addr to all nodes in its routing table and leaf-set to let them update their tables to include D_0



Node Failure/Leaving from Pastry

1) Only the node D_r whose leaf set L contains the failed node D_0 (D46A1C in Fig), can discover the failure and initiate the routing repair of D_0
2) To repair its leaf set L, D_r (D467C4 in Fig) selects a node D_1 from L that is closest to the failed node D_0, and requests the leaf set L_1 from D_1
3) D_r selects a node from L_1 whose GUID is closest to D_0 to replace D_0 in L
4) D_r then inform all other nodes in L about the failure of D_0, asking them to do the same repairing procedure as D_r



Performance of Pastry

- Routing structure is highly redundant, avoiding bottlenecks at some nodes
- All nodes send ‘heart-beat’ messages to neighboring nodes in their leaf-sets for possible failure detection and repairing
- It is evolved to MSPastry for high dependability (low msg loss rate), and later led to Tapestry system

Summary

- Purely distributed P2P and Hybrid P2P
- Napster
  - A central server for index and data files on peers
- Gnutella – a general file sharing system
  - Rely on ultrapeers to form a dense core network for routing
  - Each node (peer or ultrapeer) maintains a Query Routing Table (QRT)
- BitTorrent – a system for sharing of large video files
  - Tracker, seeder, swarm and a .torrent file
  - Tit-for-Tat and Rarest-First policy
- Pastry
  - A routing method combined of ring and tree structures
  - Each node keeps a routing table of 32 rows (128bits ID) × 15 columns

Exercises

- P2P system relies on clients to provide services to others. In BitToerrent, how does it encourage clients to do so?
- One good feature of Pastry is that it has bounded number of hops to locate an object. What is the maximal number of hops that Pastry locates an object? Explain why.
- Briefly explain the steps of a new node joining Pastry. Why the 1st row of the routing table of the new node is the same as the node who has the smallest internet delay from it?