

CSE 486/586 Distributed Systems Peer-to-Peer Architecture --- 1

Steve Ko
Computer Sciences and Engineering
University at Buffalo

CSE 486/586

Last Time

- Two multicast algorithms for total ordering
 - Sequencer
 - ISIS
- Multicast for causal ordering
 - Uses vector timestamps

CSE 486/586

2

Today's Question

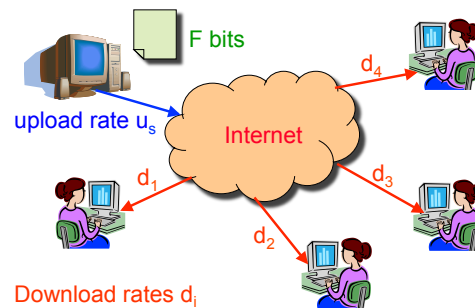
- How do we organize the nodes in a distributed system?
- Up to the 90's
 - Prevalent architecture: **client-server** (or master-slave)
 - **Unequal** responsibilities
- Now
 - Emerged architecture: **peer-to-peer**
 - **Equal** responsibilities
- Today: studying **peer-to-peer as a paradigm** (not just as a file-sharing application, but will still use file-sharing as the main example)
 - Learn the techniques and principles

CSE 486/586

3

Motivation: Distributing a Large File

- A client-server architecture can do it...



CSE 486/586

4

Motivation: Distributing a Large File

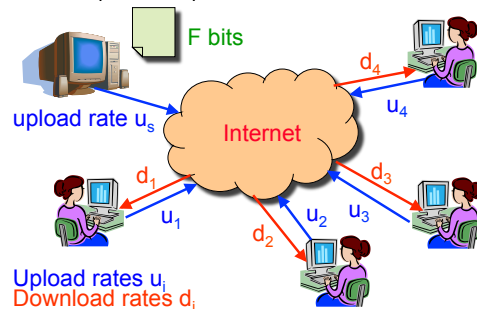
- ...but **sometimes not good enough**.
 - Limited bandwidth
 - One server can only serve so many clients.
- **Increase the upload rate** from the server-side?
 - Higher link bandwidth at the one server
 - Multiple servers, each with their own link
 - Requires deploying more infrastructure
- Alternative: **have the receivers help**
 - Receivers get a copy of the data
 - And then redistribute the data to other receivers
 - To reduce the burden on the server

CSE 486/586

5

Motivation: Distributing a Large File

- Peer-to-peer to help



CSE 486/586

6

Challenges of Peer-to-Peer

- **Peers come and go**
 - Peers are intermittently connected
 - May come and go at any time
 - Or come back with a different IP address
- **How to locate the relevant peers?**
 - Peers that are online right now
 - Peers that have the content you want
- **How to motivate peers to stay in system?**
 - Why not leave as soon as download ends?
 - Why bother uploading content to anyone else?
- **How to download efficiently?**
 - The faster, the better

CSE 486/586

7

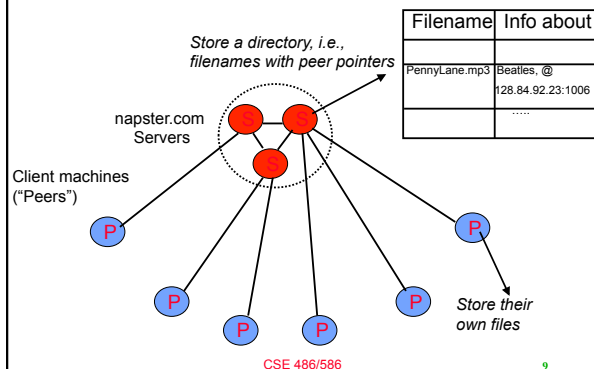
Locating Relevant Peers

- Evolution of peer-to-peer
 - **Central directory** (Napster)
 - **Query flooding** (Gnutella)
 - **Hierarchical overlay** (Kazaa, modern Gnutella)
- Design goals
 - Scalability
 - Simplicity
 - Robustness
 - Plausible deniability

CSE 486/586

8

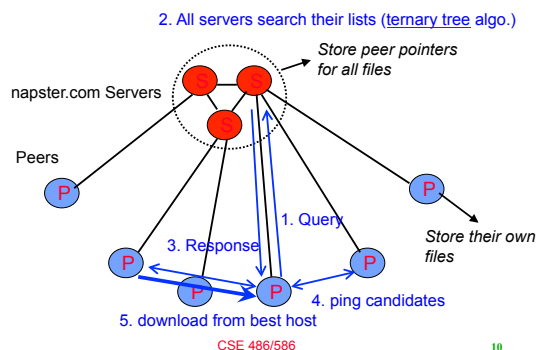
The First: Napster



CSE 486/586

9

The First: Napster



CSE 486/586

10

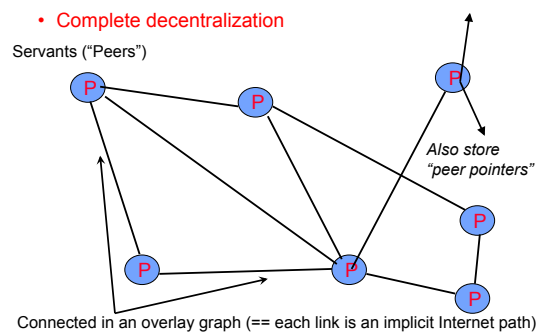
The First: Napster

- **Server's directory continually updated**
 - Always know what file is currently available
 - Point of vulnerability for legal action
- **Peer-to-peer file transfer**
 - No load on the server
 - Plausible deniability for legal action (but not enough)
- **Proprietary protocol**
 - Login, search, upload, download, and status operations
 - No security: cleartext passwords and other vulnerability
- **Bandwidth issues**
 - Suppliers ranked by apparent bandwidth & response time
- **Limitations:**
 - Decentralized file transfer, but centralized lookup

CSE 486/586

11

The Second: Gnutella

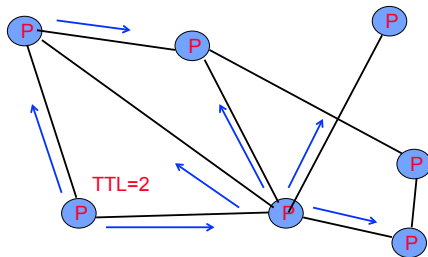


CSE 486/586

12

The Second: Gnutella

Query's flooded out, ttl-restricted, forwarded only once

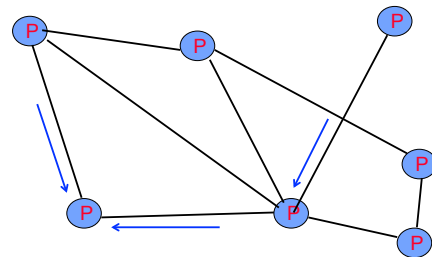


CSE 486/586

13

The Second: Gnutella

Successful results QueryHit's routed on reverse path



CSE 486/586

14

The Second: Gnutella

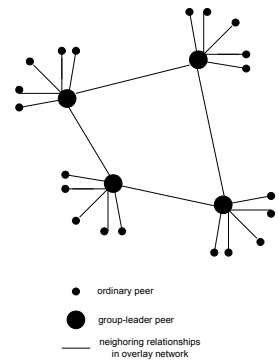
- **Advantages**
 - Fully decentralized
 - Search cost distributed
 - Processing per node permits powerful search semantics
- **Disadvantages**
 - Search scope may be quite large
 - Search time may be quite long
 - High overhead, and nodes come and go often

CSE 486/586

15

The Third: KaZA

- **Middle ground** between Napster & Gnutella
- Each peer is **either a group leader (super peer)** or **assigned to a group leader**
 - TCP connection between peer and its group leader
 - TCP connections between some pairs of group leaders
- Group leader tracks the content in all its children



CSE 486/586

16

The Third: KaZA

- A supernode stores a **directory listing** (<filename,peer pointer>), similar to Napster servers
- Supernode membership changes over time
- Any peer can become (and stay) a supernode, provided it has earned enough **reputation**
 - Kazaalite: participation level (=reputation) of a user between 0 and 1000, initially 10, then affected by length of periods of connectivity and total number of uploads
 - More sophisticated reputation schemes invented, especially based on economics
- A peer searches by contacting a nearby supernode

CSE 486/586

17

CSE 486/586 Administrivia

- Please start PA2-B.
- (In class) Midterm: 3/9 (Wednesday)

CSE 486/586

18

Now: BitTorrent

- Key motivation: **popular content**
 - Popularity exhibits temporal locality (Flash Crowds)
 - E.g., Slashdot/Digg effect, CNN Web site on 9/11, release of a new movie or game
- Focused on **efficient fetching, not searching**
 - Distribute same file to many peers
 - Single publisher, many downloaders
- Preventing free-loading

CSE 486/586

19

Key Feature: Parallel Downloading

- Divide large file into many pieces
 - **Replicate** different pieces on different peers
 - A peer with a complete piece can trade with other peers
 - Peer can (hopefully) assemble the entire file
- Allows **simultaneous downloading**
 - Retrieving different **parts** of the file from different peers **at the same time**
 - And uploading parts of the file to peers
 - Important for very large files
- System Components
 - Web server
 - Tracker
 - Peers

CSE 486/586

20

Tracker

- Infrastructure node
 - Keeps track of peers participating in the torrent
- Peers register with the tracker
 - Peer registers when it arrives
 - Peer periodically informs tracker it is still there
- Tracker selects peers for downloading
 - Returns a random set of peers
 - Including their IP addresses
 - So the new peer knows who to contact for data
- Can be "trackerless" using DHT

CSE 486/586

21

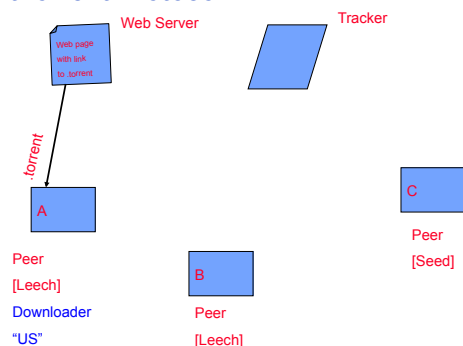
Chunks

- Large file divided into smaller pieces
 - Fixed-sized chunks
 - Typical chunk size of 256 Kbytes
- Allows **simultaneous transfers**
 - Downloading chunks from different neighbors
 - Uploading chunks to other neighbors
- Learning what chunks your neighbors have
 - Periodically asking them for a list
- File done when all chunks are downloaded

CSE 486/586

22

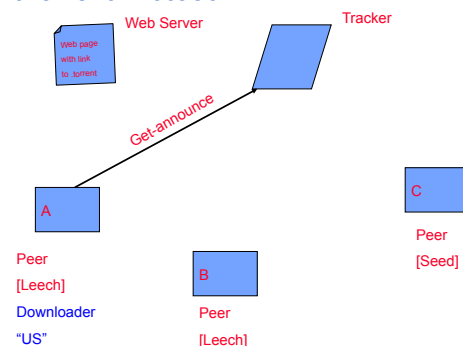
BitTorrent Protocol



CSE 486/586

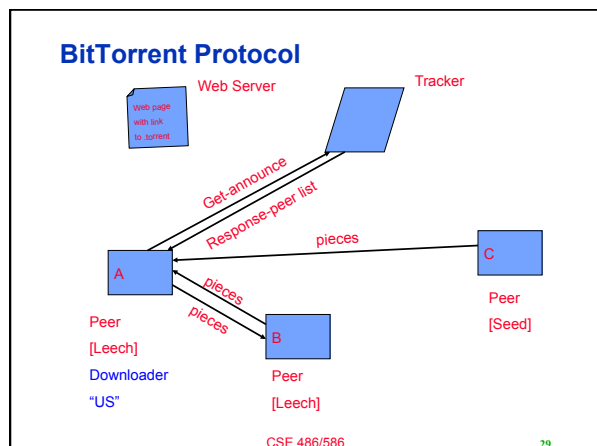
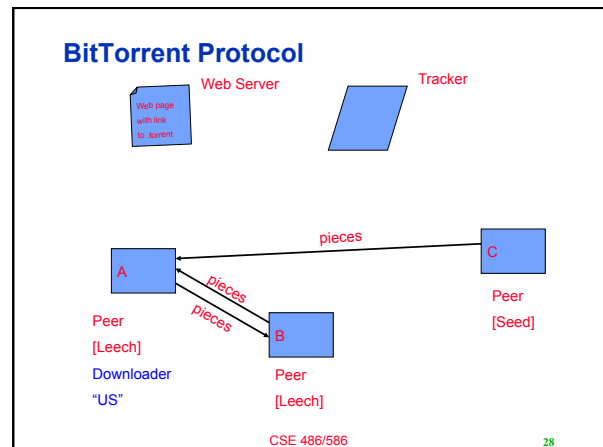
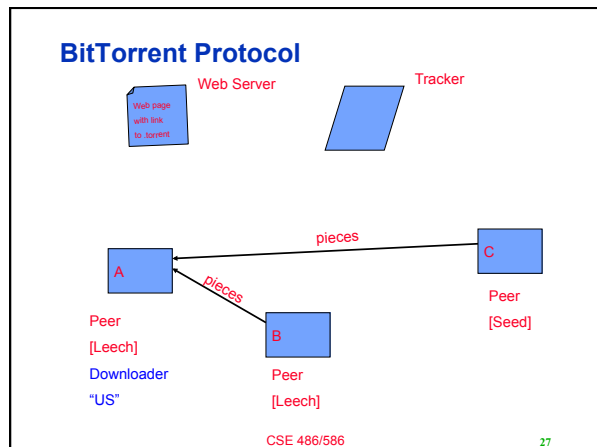
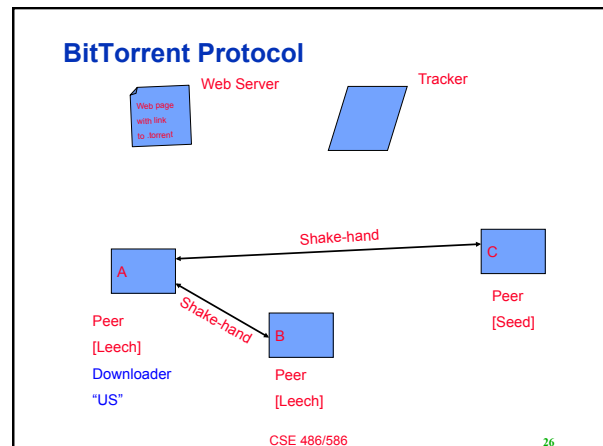
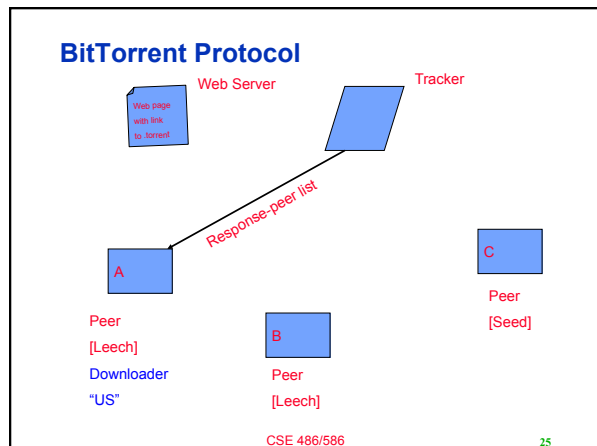
23

BitTorrent Protocol



CSE 486/586

24



Chunk Request Order

- Which chunks to request?
 - Could download in order
 - Like an HTTP client does
- Problem: many peers have the early chunks
 - Peers have little to share with each other
 - Limiting the scalability of the system
- Problem: eventually nobody has rare chunks
 - E.g., the chunks need the end of the file
 - Limiting the ability to complete a download
- Solutions: random selection and rarest first

CSE 486/586 30

Rarest Chunk First

- Which chunks to request first?
 - The chunk with the fewest available copies
 - I.e., the rarest chunk first
- Benefits to the peer
 - Avoid starvation when some peers depart
- Benefits to the system
 - Avoid starvation across all peers wanting a file
 - Balance load by equalizing # of copies of chunks

CSE 486/586

31

Preventing Free-Riding

- Vast majority of users are free-riders
 - Most share no files and answer no queries
 - Others limit # of connections or upload speed
- A few “peers” essentially act as servers
 - A few individuals contributing to the public good
 - Making them hubs that basically act as a server
- BitTorrent prevent free riding
 - Allow the fastest peers to download from you
 - Occasionally let some free loaders download

CSE 486/586

32

Preventing Free-Riding

- Peer has limited upload bandwidth
 - And must share it among multiple peers
- Prioritizing the upload bandwidth: tit for tat
 - Favor neighbors that are uploading at highest rate
- Rewarding the top four neighbors
 - Measure download bit rates from each neighbor
 - Reciprocates by sending to the top four peers
 - Recompute and reallocate every 10 seconds
- Optimistic unchoking
 - Randomly try a new neighbor every 30 seconds
 - So new neighbor has a chance to be a better partner

CSE 486/586

33

Gaming BitTorrent

- BitTorrent can be gamed, too
 - Peer uploads to top N peers at rate $1/N$
 - E.g., if $N=4$ and peers upload at 15, 12, 10, 9, 8, 3
 - ... then peer uploading at rate 9 gets treated quite well
- Best to be the N^{th} peer in the list, rather than 1st
 - Offer just a bit more bandwidth than the low-rate peers
 - But not as much as the higher-rate peers
 - And you'll still be treated well by others
- BitTyrant software
 - Uploads at higher rates to higher-bandwidth peers
 - <http://bittyrant.cs.washington.edu/>

CSE 486/586

34

BitTorrent Today

- Significant fraction of Internet traffic
 - Estimated at 30%
 - Though this is hard to measure
- Problem of incomplete downloads
 - Peers leave the system when done
 - Many file downloads never complete
 - Especially a problem for less popular content
- Still lots of legal questions remains
- Further need for incentives

CSE 486/586

35

Summary

- Evolution of peer-to-peer
 - Central directory (Napster)
 - Query flooding (Gnutella)
 - Hierarchical overlay (Kazaa, modern Gnutella)
- BitTorrent
 - Focuses on parallel download
 - Prevents free-riding
- Next: Distributed Hash Tables

CSE 486/586

36

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC), Michael Freedman (Princeton), and Jennifer Rexford (Princeton).