Indranil Gupta (Indy)
September 26, 2013
Lecture 10
Peer-to-peer Systems I

Reading: Gnutella paper on website
Why Study Peer to peer (P2P) systems?

• To understand how they work
• To understand the techniques and principles within them
• To modify, adapt, reuse these techniques and principles in other related areas
  – Cloud computing: key-value stores borrow heavily from p2p systems
  – To build your own p2p system
• To grow the body of knowledge about distributed systems
Some Questions

• Why do people get together?
  – to share information
  – to share and exchange resources they have
    • books, class notes, experiences, videos, music cd’s

• How can computers help people
  – find information
  – find resources
  – exchange and share resources
• Existing technologies: The Web!
  – Search engines
  – Forums: chat rooms, blogs, ebay
  – Online business
• But, the web is heavy weight if you want specific resources: say a Beatles’ song “PennyLane”
• A search engine will give you their bio, lyrics, chords, articles on them, and then perhaps the mp3
• But you want only the song, nothing else!
• If you can find a peer who has a copy of the Beatles song (mp3), perhaps in exchange for your UIUC Homecoming videos, that would be great!
  – Napster: a solution light weight that was lighter than the Web
### Filenames

<table>
<thead>
<tr>
<th>Filename</th>
<th>Filesize</th>
<th>Bitrate</th>
<th>Freq</th>
<th>Length</th>
<th>User</th>
<th>Connection</th>
<th>Ping</th>
</tr>
</thead>
<tbody>
<tr>
<td>incomplete_other_artist\Tito Puente Golden Latin Jazz Allstars - Oye Como ...</td>
<td>3,696,640</td>
<td>128</td>
<td>44100</td>
<td>3.51</td>
<td>bdenzler</td>
<td>DSL</td>
<td>343</td>
</tr>
<tr>
<td>incomplete_other_artist[Marty Robbins] The Fastest Gun Around.mp3</td>
<td>542,304</td>
<td>128</td>
<td>44100</td>
<td>0.39</td>
<td>bdenzler</td>
<td>DSL</td>
<td>343</td>
</tr>
<tr>
<td>incomplete_other_artist\Ravi Shankar - Chants Of India 04 - Asato Maa.mp3</td>
<td>2,449,408</td>
<td>128</td>
<td>44100</td>
<td>2.35</td>
<td>bdenzler</td>
<td>DSL</td>
<td>343</td>
</tr>
<tr>
<td>other_artist\Engelbert Humperdinck - White Christmas.mp3</td>
<td>9,277,648</td>
<td>320</td>
<td>44100</td>
<td>3.52</td>
<td>bdenzler</td>
<td>DSL</td>
<td>343</td>
</tr>
<tr>
<td>other_artist\Grateful Dead - Franklin's Tower - Reggae Style.mp3</td>
<td>4,635,458</td>
<td>128</td>
<td>44100</td>
<td>4.48</td>
<td>bdenzler</td>
<td>DSL</td>
<td>343</td>
</tr>
<tr>
<td>Unknown Artist - You seriously have to listen to this.mp3</td>
<td>462,848</td>
<td>318</td>
<td>16000</td>
<td>0.17</td>
<td>sam113...</td>
<td>Cable</td>
<td>383</td>
</tr>
<tr>
<td>MP3\artist - 'The Way Life Is' By Drag-On featuring Case.mp3</td>
<td>4,726,784</td>
<td>128</td>
<td>44100</td>
<td>4.54</td>
<td>burg651</td>
<td>Cable</td>
<td>386</td>
</tr>
<tr>
<td>MP3\artist - 'Opposite Of H2O' By Drag-On featuring Jadakiss.mp3</td>
<td>3,540,992</td>
<td>128</td>
<td>44100</td>
<td>4.31</td>
<td>burg651</td>
<td>Cable</td>
<td>386</td>
</tr>
<tr>
<td>Various Artist - Perfect Day 97.mp3</td>
<td>3,722,344</td>
<td>128</td>
<td>44100</td>
<td>3.53</td>
<td>falkstad</td>
<td>ISDN-128K</td>
<td>398</td>
</tr>
<tr>
<td>Liszt\Listz - Etude 'Un sospiro' - Czilka-artist.mp3</td>
<td>2,752,512</td>
<td>128</td>
<td>44100</td>
<td>2.53</td>
<td>Iskjdlkl...</td>
<td>Unknown</td>
<td>504</td>
</tr>
<tr>
<td>Music\Waiting To Exhale - Original Soundtrack Album - Various Artist - Count...</td>
<td>3,199,083</td>
<td>96</td>
<td>44100</td>
<td>4.26</td>
<td>Jzfork9</td>
<td>56K</td>
<td>511</td>
</tr>
<tr>
<td>Track 03_artist.mp3</td>
<td>4,054,332</td>
<td>128</td>
<td>44100</td>
<td>4.13</td>
<td>immusic...</td>
<td>Cable</td>
<td>514</td>
</tr>
<tr>
<td>Track 02_artist.mp3</td>
<td>6,228,974</td>
<td>128</td>
<td>44100</td>
<td>6.26</td>
<td>immusic...</td>
<td>Cable</td>
<td>514</td>
</tr>
<tr>
<td>Track 01_artist.mp3</td>
<td>4,731,426</td>
<td>128</td>
<td>44100</td>
<td>4.54</td>
<td>immusic...</td>
<td>Cable</td>
<td>514</td>
</tr>
<tr>
<td>Track 04_artist.mp3</td>
<td>4,514,505</td>
<td>128</td>
<td>44100</td>
<td>4.41</td>
<td>immusic...</td>
<td>Cable</td>
<td>514</td>
</tr>
<tr>
<td>Track 05_artist.mp3</td>
<td>4,105,323</td>
<td>128</td>
<td>44100</td>
<td>4.16</td>
<td>immusic...</td>
<td>Cable</td>
<td>514</td>
</tr>
<tr>
<td>mixer in track 01_Artist_0721011750.mp3</td>
<td>180,686</td>
<td>128</td>
<td>44100</td>
<td>0.17</td>
<td>immusic...</td>
<td>Cable</td>
<td>514</td>
</tr>
<tr>
<td>Album\Reflex - Keep In Touch-Artist.mp3</td>
<td>7,041,024</td>
<td>160</td>
<td>44100</td>
<td>5.49</td>
<td>rotimcga</td>
<td>56K</td>
<td>527</td>
</tr>
</tbody>
</table>

Returned 100 results.
A Brief History

- [6/99] Shawn Fanning (freshman Northeastern U.) releases Napster online music service
- [12/99] RIAA sues Napster, asking $100K per download
- [3/00] 25% UWisc traffic Napster, many universities ban it
- [00] 60M users
- [2/01] US Federal Appeals Court: users violating copyright laws, Napster is abetting this
- [9/01] Napster decides to run paid service, pay % to songwriters and music companies
- [Today] Napster protocol is open, people free to develop opennap clients and servers

http://opennap.sourceforge.net
Napster Structure

*Store a directory, i.e., filenames with peer pointers*

napster.com Servers

Client machines ("Peers")

<table>
<thead>
<tr>
<th>Filename</th>
<th>Info about</th>
</tr>
</thead>
<tbody>
<tr>
<td>PennyLane.mp3</td>
<td>Beatles, @128.84.92.23:1006</td>
</tr>
</tbody>
</table>
Napster Operations

Client

• Connect to a Napster server
• Upload list of music files that you want to share
  – Server maintains list of <filename, ip_address, portnum> tuples. Server stores no files.
• Search
  – Send server keywords to search with
  – (Server searches its list with the keywords)
  – Server returns a list of hosts - <ip_address, portnum> tuples - to client
  – Client pings each host in the list to find transfer rates
  – Client fetches file from best host
• All communication uses TCP
Napster Search

1. Query
2. All servers search their lists (ternary tree algo.)
   Store peer pointers for all files
3. Response
4. ping candidates
5. download from best host

napster.com Servers

Peers

Store their own files
Problems

• Centralized server a source of congestion
• Centralized server single point of failure
• No security: plaintext messages and passwds
• Courts declared napster.com responsible for users’ copyright violation
  – “Indirect infringement”
Gnutella

- Eliminate the servers
- Client machines search and retrieve amongst themselves
- Clients act as servers too, called servents
- [3/00] release by AOL, 88K users by 3/03
- Original design underwent several modifications
- Available as an open protocol today

http://www.limewire.com
Gnutella

Servents (“Peers”)

Connected in an **overlay** graph
(== each link is an implicit Internet path)

Store their own files

Also store “peer pointers”
How do I search for my Beatles file?

- Gnutella *routes* different messages within the overlay graph
- Gnutella protocol has 5 main message types
  - Query (search)
  - QueryHit (response to query)
  - Ping (to probe network for other peers)
  - Pong (reply to ping, contains address of another peer)
  - Push (used to initiate file transfer)
- We’ll go into the message structure and protocol now (note: all fields except IP address are in little-endian format)
Descriptor Header

<table>
<thead>
<tr>
<th>Descriptor ID</th>
<th>Payload descriptor</th>
<th>TTL</th>
<th>Hops</th>
<th>Payload length</th>
</tr>
</thead>
</table>

- ID of this search transaction
- Type of payload
  - 0x00 Ping
  - 0x01 Pong
  - 0x40 Push
  - 0x80 Query
  - 0x81 Queryhit

- Decremented at each hop
- Message dropped when ttl=0
- ttl_initial usually 7 to 10

Payload

- Number of bytes in message following this header
- Incremented at each hop

Gnutella Message Header Format
<table>
<thead>
<tr>
<th>Minimum Speed</th>
<th>Search criteria (keywords)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Query (0x80)**

Payload Format in Gnutella **Query Message**
Gnutella Search

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

Who has PennyLane.mp3?

TTL=2
**QueryHit** (0x81): successful result to a query

```
<table>
<thead>
<tr>
<th>Num. hits</th>
<th>port</th>
<th>ip_address</th>
<th>speed</th>
<th>(fileindex, filename, fsize)</th>
<th>servent_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(n, n+16)</td>
<td></td>
</tr>
</tbody>
</table>
```

**Info about responder**: Unique identifier of responder; a function of its IP address

**Payload Format in Gnutella Query Reply Message**
Gnutella Search

Successful results QueryHit’s routed on reverse path

Who has PennyLane.mp3?
Avoiding excessive traffic

To avoid duplicate transmissions, each peer maintains a list of recently received messages

- Query forwarded to all neighbors except peer from which received
- Each Query (identified by DescriptorID) forwarded only once
- QueryHit routed back only to peer from which Query received with same DescriptorID
  - If neighbor does not exist anymore, drop QueryHit
- Duplicates with same DescriptorID and Payload descriptor (msg type) are dropped
- QueryHit with DescriptorID for which Query was not seen is dropped
After receiving QueryHit messages

• Requestor chooses “best” QueryHit responder
  – Initiates HTTP request directly to responder’s ip+port
    
    GET /get/<File Index>/<File Name>/HTTP/1.0
    Connection: Keep-Alive
    Range: bytes=0-
    User-Agent: Gnutella

• Responder then replies following start message, followed by packets containing file:
  
  HTTP 200 OK
  Server: Gnutella
  Content-type:application/binary
  Content-length: 1024

• HTTP is the file transfer protocol. Why?
• Why the “range” field in the GET request?
• What if responder is behind firewall that disallows incoming connections?
Dealing with Firewalls

Requestor sends **Push** to responder asking for file transfer

Has PennyLane.mp3
But behind firewall

(Why is the Push routed and not sent directly?)
Push (0x40)

<table>
<thead>
<tr>
<th>servent_id</th>
<th>fileindex</th>
<th>ip_address</th>
<th>port</th>
</tr>
</thead>
</table>

- same as in received QueryHit
- Address at which requestor can accept incoming connections
• Responder establishes a TCP connection at ip_address, port specified. Sends

    GIV <File Index>:<Servent Identifier>/<File Name>\n\n• Requestor then sends GET to responder (as before) and file is transferred

• What if requestor is behind firewall too?
  – Gnutella gives up
  – Can you think of an alternative solution?
Ping-Pong

**Ping (0x00)**
no payload

**Pong (0x01)**

<table>
<thead>
<tr>
<th>Port</th>
<th>ip_address</th>
<th>Num. files shared</th>
<th>Num. KB shared</th>
</tr>
</thead>
</table>

- P2P systems have **churn** – peers continuously joining, leaving, and failing
- Peers initiate Ping’s periodically
- Ping’s flooded out like Query’s, Pong’s routed along reverse path (like QueryHit’s)
- Pong replies used to update set of neighboring peers
  - to keep neighbor lists fresh in spite of churn
Gnutella Summary

• No servers
• Peers/servents maintain “neighbors”, this forms an overlay graph
• Peers store their own files
• Queries flooded out, ttl restricted
• QueryHit (replies) reverse path routed
• Supports file transfer through firewalls
• Periodic Ping-pong to continuously refresh neighbor lists
  – List size specified by user at peer : heterogeneity means some peers may have more neighbors
  – Gnutella found to follow **power law** distribution:
    \[ P(\text{#links} = L) \sim L^{-k} \]  \( (k \text{ is a constant}) \)
Problems

• Ping/Pong constituted 50% traffic
  – Solution: Multiplex, *cache* and reduce frequency of pings/pongs

• Repeated searches with same keywords
  – Solution: *Cache* Query, QueryHit messages

• Modem-connected hosts do not have enough bandwidth for passing Gnutella traffic
  – Solution: use a central server to act as proxy for such peers
  – Another solution:
    ➔ FastTrack System (in a few slides)
Problems (contd.)

• Large number of *freeloaders*
  – 70% of users in 2000 were freeloaders
  – Only download files, never upload own files
  – Endemic to all p2p systems in deployment

• Flooding causes excessive traffic
  – Is there some way of maintaining meta-information about peers that leads to more intelligent routing?

→ Structured Peer-to-peer systems
  e.g., Chord System (next lecture)
FastTrack

• Hybrid between Gnutella and Napster
• Takes advantage of “healthier” participants in the system
• Underlying technology in Kazaa, KazaaLite, Grokster
• Proprietary protocol, but some details available
• Like Gnutella, but with some peers designated as supernodes
A FastTrack-like System

Peers

Supernodes
FastTrack (contd.)

• A supernode stores a directory listing (<filename,peer pointer>), similar to Napster servers
• A peer searches by contacting a nearby supernode
• 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.
BitTorrent – A Quick Overview

1. Get tracker
2. Get peers
3. Get file blocks

Website links to .torrent
Tracker, per file (receives heartbeats, joins and leaves from peers)
Peer (seed, has full file)
Peer (new, leecher)
Peer (new, leecher)
Peer (seed, has some blocks)
BitTorrent – A Quick Overview (2)

- File split into blocks (32 KB – 256 KB)
- Download **Local Rarest First** block policy: prefer early download of blocks that are least replicated among neighbors
  - Exception: New node allowed to pick one random neighbor: helps in bootstrapping
- **Tit for tat** bandwidth usage: Provide blocks to neighbors that provided it the best download rates
  - Incentive for nodes to provide good download rates
  - Both Leechers and Seeds
- **Choking**: Limit number of neighbors to which concurrent uploads <= a number (5), i.e., the “best” neighbors
  - Everyone else choked
  - Periodically re-evaluate this set (e.g., 10 s)
  - **Optimistic unchoke**: periodically (e.g., ~30 s), unchoke a random neighbor – helps keep unchoked set fresh
Wrap-up Notes

Applies to all p2p systems

• How does a peer join the system
  – Send an http request to well-known url for that P2P service - http://www.myp2pservice.com
  – Message routed (after DNS lookup) to a well known server which then initializes new peers’ neighbor table
  – Server only maintains a partial list of online clients
Summary

• Napster: protocol overview, more details available on webpage
• Gnutella protocol
• FastTrack protocol
• BitTorrent
• Protocols continually evolving, software for new clients and servers conforming to respective protocols: developer forums at
  – Napster: http://opennap.sourceforge.net
  – Gnutella: http://www.limewire.com
• Others
  – Peer to peer working groups: http://p2p.internet2.edu
For Next Lecture

• Read “Chord” paper from website
  – Sections 1-4, 6-7

• MP2
  – By now you should have a design for, and have started coding the failure detector

• Blue Waters Datacenter Tour!