BitTorrent

Kevin Kaminski
CISC856
11/27/2012
P2P In General

• Various protocols (Gnutella, eDonkey, Fastrack)
  – BitTorrent most popular
    • Next is Gnutella, though declining due to LimeWire takedown and FrostWire drop

• Various clients
  – uTorrent seems to be best/most popular
  – Vuze, eMule, Frostwire other alternatives
## Worldwide Traffic Share

### Asia-Pacific

<table>
<thead>
<tr>
<th>Application</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitTorrent</td>
<td>27.19%</td>
</tr>
<tr>
<td>YouTube</td>
<td>14.94%</td>
</tr>
<tr>
<td>HTTP</td>
<td>10.44%</td>
</tr>
<tr>
<td>PPStream</td>
<td>6.36%</td>
</tr>
<tr>
<td>Thunder</td>
<td>4.62%</td>
</tr>
<tr>
<td>Flash Video</td>
<td>3.36%</td>
</tr>
<tr>
<td>QVoD</td>
<td>3.44%</td>
</tr>
<tr>
<td>Facebook</td>
<td>2.08%</td>
</tr>
<tr>
<td>iTunes</td>
<td>1.86%</td>
</tr>
<tr>
<td>MPEG Streaming</td>
<td>1.61%</td>
</tr>
<tr>
<td><strong>Top 10</strong></td>
<td>75.08%</td>
</tr>
</tbody>
</table>

### Europe

<table>
<thead>
<tr>
<th>Application</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitTorrent</td>
<td>20.32%</td>
</tr>
<tr>
<td>HTTP</td>
<td>17.70%</td>
</tr>
<tr>
<td>YouTube</td>
<td>15.25%</td>
</tr>
<tr>
<td>eDonkey</td>
<td>9.39%</td>
</tr>
<tr>
<td>Flash Video</td>
<td>4.70%</td>
</tr>
<tr>
<td>RTMP</td>
<td>2.47%</td>
</tr>
<tr>
<td>Facebook</td>
<td>2.43%</td>
</tr>
<tr>
<td>SSL</td>
<td>1.74%</td>
</tr>
<tr>
<td>MPEG Streaming</td>
<td>1.66%</td>
</tr>
<tr>
<td><strong>Top 10</strong></td>
<td>77.19%</td>
</tr>
</tbody>
</table>

### Latin America

<table>
<thead>
<tr>
<th>Application</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>26.61%</td>
</tr>
<tr>
<td>HTTP</td>
<td>19.98%</td>
</tr>
<tr>
<td>BitTorrent</td>
<td>9.24%</td>
</tr>
<tr>
<td>Facebook</td>
<td>6.97%</td>
</tr>
<tr>
<td>Flash Video</td>
<td>5.83%</td>
</tr>
<tr>
<td>Skype</td>
<td>3.93%</td>
</tr>
<tr>
<td>Ares</td>
<td>2.77%</td>
</tr>
<tr>
<td>MPEG Streaming</td>
<td>2.04%</td>
</tr>
<tr>
<td>Windows Update</td>
<td>1.85%</td>
</tr>
<tr>
<td>SSL</td>
<td>1.45%</td>
</tr>
<tr>
<td><strong>Top 10</strong></td>
<td>80.67%</td>
</tr>
</tbody>
</table>

- 2012 statistics courtesy of Sandvine
Server Client Model

• Centralized
• Fast speeds
• Stable
P2P Model

• Each node is a “peer”
• Peers both clients and servers
• Decentralized
Why P2P?

- **Advantages**
  - Servers require expensive hardware, P2P can be used on common desktops
  - Redundancy - Resources located in multiple locations, no single point of failure

- **Disadvantages**
  - Slower speeds near endpoints
  - Peers constantly connecting/disconnecting, unstable
  - If no one sharing the resource, cannot be downloaded
BitTorrent Component Overview

• Target file
  – Broken into indexed “pieces”

• Metainfo file
  – Hosted on BitTorrent index, contains meta information about the torrent file

• Tracker
  – Centralized server that coordinates downloads

• Peer Wire Protocol
  – Used to transfer pieces between peers
General Process

1. Download .torrent file
2. Contact tracker
3. Download peer list
4. Connect to peers and download file pieces
5. Upload concurrently to peers
Metainfo File

- Hosted on web server (torrent index) and downloaded out-of-band
- Contains tracker and file information in “dictionary” of keys
- Dictionary maps one value to another
- File is encoded using “bencoding”
d8:announce39:http://torrent.ubuntu.com:6969/announce
l44:http://ipv6.torrent.ubuntu.com:6969/announceeee
7:comment33:Torrent downloaded from 1337x.org
10:created by8:atik0786
13:creation datei1335433885e
4:info
d6:lengthi735358976e
4:name29:ubuntu-12.04-desktop-i386.iso
12:piece lengthi524288e
6:pieces28060:.............ee
Bencoding

- **Integers**: $i \ <\text{number}\ >\ e$
  - $i404e$ is the integer 404

- **Strings**: $<\text{string length}> : <\text{string data}>$
  - $7:example$ is the string “example”

- **Lists**: $l <\text{bencoded values}> e$
  - $l7:examplei404ee$ is [“example”, 404]

- **Dictionaries**: $d <\text{string}><\text{bencoded value}> e$
  - $d5:alpha1:a4:beta1:be$ is [“alpha”:”a”, “beta”:”b”]
Metainfo File Keys

**announce**
http://torrent.ubuntu.com:6969/announce

**announce-list**
http://torrent.ubuntu.com:6969/announce
http://ipv6.torrent.ubuntu.com:6969/announce

**comment**
Torrent downloaded from 1337x.org

**created by**
atik0786

**creation date**
1335433885

**info**

- **length**
  735358976

- **name**
  ubuntu-12.04-desktop-i386.iso

- **piece length**
  524288

- **pieces**
  28060:......
[ "announce": "http://torrent.ubuntu.com:6969/announce",
  "announce-list": [ [ "http://torrent.ubuntu.com:6969/announce" ],
    ["http://ipv6.torrent.ubuntu.com:6969/announce"] ],
  "comment": "Torrent downloaded from 1337x.org",
  "created by": "atik0786",
  "creation date": 1335433885,
  "info": [ "length": 735358976,
    "name": "ubuntu-12.04-desktop-i386.iso",
    "piece": 524288,
    "pieces": "<28060 byte string of concatenated 20 byte SHA-1 hashes of each piece>" ] ]
Tracker

- Coordinates the communication between peers
- Tracks statistics of torrents
- Typically a public server
  - Not all torrent indexes have a tracker
  - Public vs private trackers

<table>
<thead>
<tr>
<th>#</th>
<th>Tracker</th>
<th>Torrents</th>
<th>Peers</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PublicBitTorrent</td>
<td>2,484,145</td>
<td>21,694,091</td>
<td>Opentracker</td>
</tr>
<tr>
<td>2</td>
<td>OpenBitTorrent</td>
<td>2,388,738</td>
<td>21,186,589</td>
<td>Opentracker</td>
</tr>
<tr>
<td>3</td>
<td>Denis.Stalker</td>
<td>1,614,356</td>
<td>10,527,993</td>
<td>Opentracker</td>
</tr>
<tr>
<td>4</td>
<td>1337x.org</td>
<td>302,799</td>
<td>4,889,991</td>
<td>XBT</td>
</tr>
<tr>
<td>5</td>
<td>Torrent.to</td>
<td>326,467</td>
<td>3,205,170</td>
<td>Opentracker</td>
</tr>
</tbody>
</table>
Tracker (cont'd)

1. Download .torrent file
2. Contact tracker
3. Download peer list
4. Connect to peers and download file pieces
5. Upload concurrently to peers
Tracker Communication

```
GET /announce?info_hash=%bb%b6%dbi%96Z%f7i%f6d%b6cny%14%f8sQA%b3&
peer_id=-UT3210-%b6m%10%ea%bb%01%1eDkV%bc%e5
&port=42176&uploaded=0&downloaded=0&left=735358976&
corrupt=0&key=7F5DA749&
event=started&numwant=200&compact=1&no_peer_id=1 HTTP/1.1
Host: torrent.ubuntu.com:6969
User-Agent: uTorrent/3210(28086)
Accept-Encoding: gzip
Connection: Close
```
Tracker Response

HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 313

"peer_id": "UT3210-%b6m%10%ea%bb%01%1eDkV%bc%e5",
"ip": "192.168.1.8",
"port": "42176"
Peer Wire Protocol (PWP)

- Communication and data transfer between peers
- Requests pieces of target file in 16KB “blocks”
  - Pieces generally 512KB-2MB in size
- States
  - Choked: remote peer does not respond to requests for blocks from peer
  - Unchoked: remote peer will upload data to the peer
  - Interested: peer expresses want to request blocks from remote peer
  - Uninterested: peer will not request blocks from remote peer
Establishing Connections

- Client peer opens PWP connections with all peers in the list from the tracker

- Handshake PDU
  - **Pstrlen**: 1 byte, length of pstr
    - 19 for Bittorrent
  - **pstr**: variable length, protocol ID string
    - “BitTorrent protocol”
  - **reserved**: 8 bytes for extensions
  - **info_hash**: 20 byte SHA-1 of entire info key in metainfo file
  - **peer_id**: 20 byte ID, same as transmitted in tracker request
Peer A parses the peer list returned by the tracker:

- Peer B IP: 2000

Peer A sets `am_interested = 0`.

Connection established.

BitTorrent Protocol Handshake:
- peer_id: A

BitTorrent Protocol Handshake:
- peer_id: B

Message Exchange:
- If max_connections > connections + 1
- peer_choking = 1

Handshaking
BitTorrent Message PDU

- **Format**
  - 4 byte **length** value, 1 byte **ID** value, variable **payload**
- Multiple PDU's can be sent in a single TCP PDU
- **Types**
  - **keep-alive** – sent if no commands sent to keep connection alive
    - Connection timeout = ~2 minutes
  - **choke** (0) – local peer choking remote peer
  - **unchoke** (1) – local peer unchoking remote peer
  - **interested** (2) – local peer is interested
  - **uninterested** (3) – local peer is uninterested
BitTorrent Message PDU (cont'd)

- **Have** (4) – payload is piece index
  - Sent after downloaded and hash verified
- **Bitfield** (5) – payload is bitfield of piece indexes
  - Bit set to 1 if peer has the piece at that index
  - First message sent after handshake, optional if peer has no pieces
- **request** (6)
  - **index** – piece index
  - **begin** – byte offset of block within piece
  - **length** – length of requested block
  - Default block length is 16KB, peer drops connections for any request lengths over 32KB
BitTorrent Message PDU (cont'd)

- **piece (7)**
  - **index** – piece index
  - **begin** – byte offset of block
  - **block** – the requested block data

- **cancel (8) - index, begin, length**
  - Cancels a block request

- **port (9) - 2 byte port number**
  - Used for DHT
Message Flow

Peer A

0 1 2

Peer B

0 1 2

Bitmap 010

Have Index=1

Interested

Unchoke

Request Index=1
Begin=0x0
Length=0x4000

Piece Index=1
Begin=0x0
Block

Have Index=1
Choke Algorithm

- Local peer keeps a list of interested and uninterested remote peers
- Interested peers are ranked by their upload rate to the local peer
- The top four peers are unchoked - downloaders
  - Every 10 seconds, rates and downloaders are recalculated
    - Time cycle reduces fibrillation – rapid choking and unchoking
    - If a peer has a better upload rate than the downloaders but is uninterested, unchoke
      - If it becomes interested in the future, it replaces the downloader with the lowest upload rate
- Reduces leechers
  - in order to become un choked, must upload to peer
Optimistic Unchoke

- Every 30 seconds, one random and choked peer is unchoked regardless of its upload rate
  - If interested, counts as one of the four downloaders
    - If uninterested, unchoke and randomly select a new choked peer
- Advantages
  - Allows faster connections to be discovered
  - Selected peer may be new and have no pieces to share
    - Optimistic unchoke will give peer its first piece so it can upload to others and become a downloader
Peer Selection Process

<table>
<thead>
<tr>
<th></th>
<th>Peer</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peer D</td>
<td>0 Mb/s</td>
</tr>
<tr>
<td>2</td>
<td>Peer E</td>
<td>0 Kb/s</td>
</tr>
<tr>
<td>3</td>
<td>Peer C</td>
<td>0 Kb/s</td>
</tr>
<tr>
<td>4</td>
<td>Peer B</td>
<td>0 Kb/s</td>
</tr>
</tbody>
</table>

- Peer D optimistically unchoke E
- Peer A request 0
- Peer B request 0
- Peer C request 1
- Peer A request 1
- Peer B request 0
- Peer C request 1
- Peer D request 0

Optimistic unchoke A

Peer E optimistically unchoke A
Seeder Mode

- Entered once peer has all pieces
- Uploads to four peers, ranked by upload rate
- Optimistic unchoke
Snubbing

- Peer may become choked by all uploaders
- Anti-snubbing – local peer cannot unchoke a peer if
  - The peer has received blocks from the local peer
  - The peer has not uploaded any blocks to the local peer
- Snubbing peer is not unchoked unless by optimistic unchoke
- Allows snubbed peer to find better peers
  - Can select multiple peers for optimistic unchoke, improves recovery time
Piece Selection Algorithm

- **Random first** – randomly selects a piece to request
  - Occurs when peer has downloaded less than 4 pieces
  - Allows fast acquisition of pieces

- **Rarest first** – requests the rarest pieces first
  - Counter tracks the number of peers who have a specific piece
  - Pieces are ranked by rarity, requested in low to high order
  - Allows pieces to be have a more equal distribution

- **Strict block policy** – when a block is requested, all other requested blocks from the same piece have higher priority
  - Complete pieces faster so they can be uploaded to other peers
End Game Mode

- Last blocks of a download can be delayed
- Send block request to all peers
  - Once received, send cancel to other peers
  - Helps last blocks download faster
  - Not reliant on a single, possibly slow peer
- Usually occurs after all blocks are requested or a download threshold is reached
End Game Mode Process