Peer-to-Peer Systems

Stefano Ferretti
DISI
University of Bologna

s.ferretti@unibo.it
About me

- Stefano Ferretti
- e-mail: s.ferretti@unibo.it
- Web: http://www.cs.unibo.it/~sferrett/
  - Try to google “Stefano Ferretti”
  - You'll find these slides in the Teaching section
- Department of Computer Science and Engineering
- Via Malaguti 1D
  - Office 1
  - Map in my webpage
The Module

- 20 hours
- Info on the Web page course
## Schedule

<table>
<thead>
<tr>
<th></th>
<th>Lunedì</th>
<th>Martedì</th>
<th>Mercoledì</th>
<th>Giovedì</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:30-10:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:30-11:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:30-12:30</td>
<td></td>
<td></td>
<td></td>
<td>Lesson</td>
</tr>
<tr>
<td>12:30-13:30</td>
<td></td>
<td></td>
<td>Lesson</td>
<td></td>
</tr>
<tr>
<td>13:30-14:30</td>
<td></td>
<td></td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>14:30-15:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:30-16:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:30-17:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:30-18:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Office hours: **e-mail** required → s.ferretti@unibo.it
E-mail

- E-mail subject:
  - CS: ...
Books and References

References at the end of the slides

Interesting Books

- Peer to Peer Systems and Applications, R. Steinmetz, K. Wehrle, LNCS 3485, Springer Verlag, 2005
- Sasu Takoma, Overlay Networks, Toward Information Networking, Taylor and Francis, 2010
- Maarten Van Steen, Graph Theory and Complex Networks, Gennaio 2010
- Buford, Yu, Lua, P2P Networking and Applications, Morgan Kaufmann, 2009
Outline (of the module)

1. Introduction to P2P systems
2. File sharing, replication, piece selection, peer selection
3. BitTorrent protocol I
4. BitTorrent protocol II
5. Distributed Hash Tables, Chord
6. Other seminal structured DHT-based approaches: Pastry, CAN
7. P2P and Streaming applications, Spotify
8. NetLogo
9. Peersim
10. Project Discussion
Outline (of this lecture)

- Overview
  - What is a P2P application?
  - Popularity of P2P applications?
- Content Replication
Definition: Overlay

- Overlay Network
  - Network at the application layer (layer 7)
Definition: Overlay

- Formed by nodes communicating among themselves
  - Dedicated machines
  - End-users

- Types of overlay
  - General purpose overlay (application-layer multicast)
  - Application specific overlay (CDN)

- Overlay construction
  - Network topology
  - Network metrics (delay, bandwidth, etc.)
Definition: Overlay

Why do we need overlays?

- Create a service that is not (or that cannot be) provided by the network (layer 3)
  - Create an application layer service

Example of services

- Application layer multicast
- Content Delivery Network (CDN)
- DNS (IP only provides IP addresses and don’t know how to route on names)
Definition: Peer

Peer

- A computer, an end-user, an application, etc.
  - Depends on the context
  - Always an end system, but an end system is not always a peer
  - An end system can be a dedicated video server that is part of a CDN, or a BitTorrent client that is part of a P2P network
Definition: Peer

- **Leecher**
  - A peer that is client and server
  - In the context of content delivery
    - Has a partial copy of the content

- **Seed**
  - A peer that is only server
  - In the context of content delivery
    - Has a full copy of the content
Definition: P2P

- Peer-to-peer applications
  - No general definition
  - Application specific
    - If not, overlay is a more appropriate definition
  - At least two peers
  - Every peer is a client and a server
    - For a same application
    - Possibility of hierarchy
  - Peers form an overlay network
Definition: P2P

Overlay Network vs. P2P applications
- A P2P application forms an overlay network
- An overlay network is not always a P2P application
- Trend to define a P2P application as overlay network formed by end-users
- Depends on the definition of P2P
The case of the Web

- Service: HTML pages access
- Pages served only by dedicated machines (HTTP servers)
  - End-users cannot serve HTML pages
- No share of HTML pages among servers: servers are not communicating among themselves, but with clients
- This is not an overlay network!
Example: Email Servers

- The case of Email servers
  - Service: Email delivery
  - POP/SMTP/IMAP servers are dedicated machine
  - Email servers communicate to deliver emails
  - This is an overlay network!
  - But, not a P2P application
  - Probably the oldest example of overlay
The New P2P Paradigm

- Web, Email, etc. is an old technology
- Is overlay network an old techno?
- Yes, when applied to servers
- But, its applications to end-users is recent
  - New applications
  - New problems
  - New techniques, algorithms, protocols
  - This is P2P!
The New P2P Paradigm

- Why P2P applications became popular in mid-2000 only?
  - High speed Internet connections
  - Power shift from servers to end-users
      - Still alive (01/2006): http://lwn.net/Articles/169961/

- P2P applications are a true revolution
  - Aside TCP/IP and the Web
New P2P applications

- P2P applications capitalize on any resource from anybody
  - P2P applications can share CPU, bandwidth and storage
    - seti@home (not P2P, but distributed)
    - BitTorrent, Emule, Gnutella
    - Skype, Google talk
    - Publius
    - Spotify
P2P file sharing taxonomy

- Current taxonomy for P2P file sharing [11]
  - Unstructured P2P: BitTorrent, Gnutella, KaZaA, etc.
  - Structured: DHT (e.g., Chord, CAN, Pastry, Kademlia, etc.)

Quite reasonable classification
Unstructured P2P

- Links do **not** depend on the contents owned by nodes
- You might inspect the whole overlay
- Easy to build/maintain

*Whom I'm gonna ask for X content?*
Structured P2P

- Links *do depend* on the contents owned by nodes
- Efficient lookup strategies

*Whom I'm gonna ask for X content?*
P2P file sharing taxonomy

- Current taxonomy for P2P file sharing [11]
  - Unstructured P2P: BitTorrent, Gnutella, KaZaA, etc.
  - Structured: DHT (e.g., Chord, CAN, Pastry, Kademlia, etc.)

Quite reasonable classification

- What is (at least, might be) wrong?
  - Assume that P2P applications must be classified according to their content localization architecture
P2P file sharing taxonomy

- Proposed taxonomy for P2P file sharing
  - Content localization
    - Unstructured
    - Structured
  - Content replication
    - Parallel download
    - File splitting
    - Piece selection
      - rarest first, random, sequential, etc.
    - Peer selection
      - choke algorithm, tit-for-tat, priority queue, etc.
P2P file sharing taxonomy

Is it better?
- No more a focus on content localization
- Decouple file localization and file replication

See the discussion thread on the P2Prq
http://www1.ietf.org/mail-archive/web/p2prg/current/msg00816.html
- No agreement
- This is a complex problem (many applications of P2P from sensor networks to BitTorrent)
- Taxonomy depends on where is your expertise
- The one proposed in the previous slide is still weak
Outline

- Overview
  - What is a P2P application?
  - Popularity of P2P applications?

- Content Replication
Measurement of P2P traffic

- Hard to perform P2P traffic measurements

Known techniques [3,4]

- Port detection
  - Can be easily circumvented

- Layer 7 inspection
  - Do not work with encrypted payload
  - Database of signatures hard to maintain

- Heuristic
  - Hard to identify, similar to other applications
Dominant Traffic with Time [1]
2007 Traffic (North America)
P2P as File Sharing ... it's not only file sharing
History of BitTorrent Usage
End 2004

- Cache Logic study [1]
  - Deep packet inspection
- BitTorrent is dominating the Internet traffic
  - 30% of the internet traffic!
- Shift of demand from music to movies
- Major sources of torrent file discontinued due to legal actions (e.g., Suprnova.org)
History of BitTorrent Usage 2005

- Cache Logic study [1]
  - Deep packet inspection

- Shift from BitTorrent to eDonkey
  - eDonkey is fully decentralized
  - Many clients and localization

- But, this is not the end of the story
  - Decentralized versions of BitTorrent
  - New large BitTorrent services (ThePirateBay, mininova, isohunt, etc.)
History of BitTorrent Usage 2006 to 2008

- Unofficial numbers from FT (September 2006)
  - 80% to 90% of P2P traffic
    - Only 10% to 20% of P2P traffic due to BitTorrent
    - The rest is due to Emule
  - Traffic captured at a BRAS (Broadband Remote Access Server)
    - Aggregate several DSLAMs – DSL access multiplexers (at ISPs)
      - (thousands of peers)
    - Claimed layer 7 inspection, but no access to the data and methodology

- 78% of P2P traffic in Japan in 2008 [41]
History of BitTorrent Usage 2010

- Sandvine (Fall 2010)
  - BitTorrent is dominating P2P
    - Except in Latin America where it is Ares
    - But Ares implements BitTorrent, thus it is not clear how much Ares traffic is BitTorrent
  - P2P is dominating upstream traffic
  - Real-time entertainment (audio and video streaming) is dominating the downstream traffic
    - But, P2P share has significantly increased since 2009 (doubled in some regions)
BitTorrent Usage in 2011

- Sandvine Report [53]
- Customers
  - ISPs
- Internet traffic statistics
  - Voluntarily submitted by a representative cross-section of Sandvine’s customer base
    - In this report 220 ISPs in 85 countries
      - Hundreds of millions of subscribers
BitTorrent Usage in 2011

**Peak Period Aggregate Traffic Composition**
(North America, Fixed Access)

<table>
<thead>
<tr>
<th>Year</th>
<th>Real-Time Entertainment</th>
<th>P2P Filesharing</th>
<th>Web Browsing</th>
<th>Real-Time Communications</th>
<th>Social Networking</th>
<th>Gaming</th>
<th>Secure Tunneling</th>
<th>Outside Top 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>29.5%</td>
<td>15.1%</td>
<td>38.7%</td>
<td>4.6%</td>
<td>2.3%</td>
<td>2.7%</td>
<td>3.1%</td>
<td>9.8%</td>
</tr>
<tr>
<td>2010</td>
<td>42.7%</td>
<td>19.2%</td>
<td>20.2%</td>
<td>2.7%</td>
<td>3.1%</td>
<td>2.8%</td>
<td>2.9%</td>
<td>12.1%</td>
</tr>
<tr>
<td>March, 2011</td>
<td>49.2%</td>
<td>18.8%</td>
<td>16.6%</td>
<td>2.8%</td>
<td>2.9%</td>
<td>2.8%</td>
<td>2.9%</td>
<td>9.7%</td>
</tr>
</tbody>
</table>

*Source: Sandvine*
# BitTorrent Usage in 2011

Table 1 - North America, Fixed Access, Peak Period, Top Applications by Bytes

<table>
<thead>
<tr>
<th>Rank</th>
<th>Upstream Application</th>
<th>Upstream Share</th>
<th>Downstream Application</th>
<th>Downstream Share</th>
<th>Aggregate Application</th>
<th>Aggregate Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BitTorrent</td>
<td>52.01%</td>
<td>Netflix</td>
<td>29.70%</td>
<td>Netflix</td>
<td>24.71%</td>
</tr>
<tr>
<td>2</td>
<td>HTTP</td>
<td>8.31%</td>
<td>HTTP</td>
<td>18.36%</td>
<td>BitTorrent</td>
<td>17.23%</td>
</tr>
<tr>
<td>3</td>
<td>Skype</td>
<td>3.81%</td>
<td>YouTube</td>
<td>11.04%</td>
<td>HTTP</td>
<td>17.18%</td>
</tr>
<tr>
<td>4</td>
<td>Netflix</td>
<td>3.59%</td>
<td>BitTorrent</td>
<td>10.37%</td>
<td>YouTube</td>
<td>9.85%</td>
</tr>
<tr>
<td>5</td>
<td>PPStream</td>
<td>2.92%</td>
<td>Flash Video</td>
<td>4.88%</td>
<td>Flash Video</td>
<td>3.62%</td>
</tr>
<tr>
<td>6</td>
<td>MGCP</td>
<td>2.89%</td>
<td>iTunes</td>
<td>3.25%</td>
<td>iTunes</td>
<td>3.01%</td>
</tr>
<tr>
<td>7</td>
<td>RTP</td>
<td>2.85%</td>
<td>RTMP</td>
<td>2.92%</td>
<td>RTMP</td>
<td>2.46%</td>
</tr>
<tr>
<td>8</td>
<td>SSL</td>
<td>2.75%</td>
<td>Facebook</td>
<td>1.91%</td>
<td>Facebook</td>
<td>1.86%</td>
</tr>
<tr>
<td>9</td>
<td>Gnutella</td>
<td>2.12%</td>
<td>SSL</td>
<td>1.43%</td>
<td>SSL</td>
<td>1.68%</td>
</tr>
<tr>
<td>10</td>
<td>Facebook</td>
<td>2.00%</td>
<td>Hulu</td>
<td>1.09%</td>
<td>Skype</td>
<td>1.29%</td>
</tr>
<tr>
<td></td>
<td><strong>Top 10</strong></td>
<td><strong>83.25%</strong></td>
<td><strong>Top 10</strong></td>
<td><strong>84.95%</strong></td>
<td><strong>Top 10</strong></td>
<td><strong>82.89%</strong></td>
</tr>
</tbody>
</table>

Source: Sandvine Network Demographics
BitTorrent Usage in 2011

Peak Period Aggregate Traffic Composition
(Latin America, Mobile Access)

- January, 2010
  - Outside Top 5: 21.9%
  - Storage and Back-Up Services: 24.5%
  - Software Updates: 34.7%
  - Social Networking: 27.5%
  - P2P Filesharing: 11.1%
  - Web Browsing: 18.0%
  - Real-Time Entertainment: 18.0%

- September, 2010
  - Outside Top 5: 8.5%
  - Storage and Back-Up Services: 11.4%
  - Software Updates: 6.6%
  - Social Networking: 15.4%
  - P2P Filesharing: 26.8%
  - Web Browsing: 25.2%
  - Real-Time Entertainment: 24.9%

- March, 2011
  - Outside Top 5: 9.9%
  - Storage and Back-Up Services: 12.1%
  - Software Updates: 4.8%
  - Social Networking: 13.7%
  - P2P Filesharing: 18.0%
  - Web Browsing: 24.9%
  - Real-Time Entertainment: 18.0%
BitTorrent Usage in 2011

Table 4 - Latin America, Mobile Access, Peak Period, Top Applications by Bytes

<table>
<thead>
<tr>
<th>Rank</th>
<th>Upstream Application</th>
<th>Upstream Share</th>
<th>Downstream Application</th>
<th>Downstream Share</th>
<th>Aggregate Application</th>
<th>Aggregate Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ares</td>
<td>39.22%</td>
<td>HTTP</td>
<td>26.78%</td>
<td>HTTP</td>
<td>24.79%</td>
</tr>
<tr>
<td>2</td>
<td>HTTP</td>
<td>15.74%</td>
<td>YouTube</td>
<td>14.76%</td>
<td>Ares</td>
<td>15.48%</td>
</tr>
<tr>
<td>3</td>
<td>Facebook</td>
<td>15.41%</td>
<td>Facebook</td>
<td>12.77%</td>
<td>Facebook</td>
<td>13.25%</td>
</tr>
<tr>
<td>4</td>
<td>BitTorrent</td>
<td>3.60%</td>
<td>Ares</td>
<td>10.26%</td>
<td>YouTube</td>
<td>12.58%</td>
</tr>
<tr>
<td>5</td>
<td>YouTube</td>
<td>2.65%</td>
<td>Google Video</td>
<td>6.08%</td>
<td>Google Video</td>
<td>5.12%</td>
</tr>
<tr>
<td>6</td>
<td>Skype</td>
<td>2.35%</td>
<td>Flash Video</td>
<td>5.92%</td>
<td>Flash Video</td>
<td>5.06%</td>
</tr>
<tr>
<td>7</td>
<td>SSL</td>
<td>2.34%</td>
<td>MegaUpload</td>
<td>4.08%</td>
<td>MegaUpload</td>
<td>3.45%</td>
</tr>
<tr>
<td>8</td>
<td>MSN Messenger</td>
<td>2.03%</td>
<td>Windows Update</td>
<td>3.50%</td>
<td>Windows Update</td>
<td>2.99%</td>
</tr>
<tr>
<td>9</td>
<td>Hotmail</td>
<td>1.40%</td>
<td>Shockwave Flash</td>
<td>1.50%</td>
<td>BitTorrent</td>
<td>1.73%</td>
</tr>
<tr>
<td>10</td>
<td>Pando</td>
<td>1.33%</td>
<td>SSL</td>
<td>1.46%</td>
<td>SSL</td>
<td>1.62%</td>
</tr>
<tr>
<td>Top 10</td>
<td>86.07%</td>
<td>Top 10</td>
<td>87.11%</td>
<td>Top 10</td>
<td>86.07%</td>
<td></td>
</tr>
</tbody>
</table>

SOURCE: SANDVINE NETWORK DEMOGRAPHICS
BitTorrent Usage in 2011

Peak Period Aggregate Traffic Composition
(Europe, Fixed Access)

- 2009:
  - Outside Top 5: 12.4%
  - News Groups: 5.4%
  - Secure Tunneling: 3.9%
  - Gaming: 3.9%
  - Storage and Back-Up Services: 25.9%
  - Real-Time Communications: 22.0%
  - Bulk Entertainment: 30.4%
  - P2P Filesharing: 31.9%
  - Web Browsing: 30.4%
  - Real-Time Entertainment: 33.2%

- 2010:
  - Outside Top 5: 6.3%
  - News Groups: 2.3%
  - Secure Tunneling: 3.9%
  - Gaming: 2.3%
  - Storage and Back-Up Services: 44.6%
  - Real-Time Communications: 11.0%
  - Bulk Entertainment: 31.9%
  - P2P Filesharing: 33.2%
  - Web Browsing: 18.0%

- March, 2011:
  - Outside Top 5: 12.3%
  - News Groups: 3.5%
  - Secure Tunneling: 3.1%
  - Gaming: 3.1%
  - Storage and Back-Up Services: 30.1%
  - Real-Time Communications: 30.1%
  - Bulk Entertainment: 33.2%
  - P2P Filesharing: 33.2%
  - Web Browsing: 18.0%

sandvine
Intelligent Broadband Networks
### BitTorrent Usage in 2011

Table 6 - Europe, Fixed Access, Peak Period, Top Applications by Bytes

<table>
<thead>
<tr>
<th>Rank</th>
<th>Application</th>
<th>Upstream Share</th>
<th>Application</th>
<th>Downstream Share</th>
<th>Application</th>
<th>Aggregate Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BitTorrent</td>
<td>59.68%</td>
<td>BitTorrent</td>
<td>21.63%</td>
<td>BitTorrent</td>
<td>28.40%</td>
</tr>
<tr>
<td>2</td>
<td>Skype</td>
<td>7.16%</td>
<td>HTTP</td>
<td>20.47%</td>
<td>HTTP</td>
<td>18.08%</td>
</tr>
<tr>
<td>3</td>
<td>HTTP</td>
<td>7.02%</td>
<td>YouTube</td>
<td>14.13%</td>
<td>YouTube</td>
<td>11.93%</td>
</tr>
<tr>
<td>4</td>
<td>PPStream</td>
<td>3.64%</td>
<td>RTMP</td>
<td>4.58%</td>
<td>RTMP</td>
<td>3.90%</td>
</tr>
<tr>
<td>5</td>
<td>Spotify</td>
<td>2.91%</td>
<td>Flash Video</td>
<td>3.99%</td>
<td>Flash Video</td>
<td>3.38%</td>
</tr>
<tr>
<td>6</td>
<td>SSL</td>
<td>2.66%</td>
<td>iTunes</td>
<td>3.65%</td>
<td>SSL</td>
<td>3.09%</td>
</tr>
<tr>
<td>7</td>
<td>eDonkey</td>
<td>1.76%</td>
<td>SSL</td>
<td>3.18%</td>
<td>iTunes</td>
<td>3.07%</td>
</tr>
<tr>
<td>8</td>
<td>YouTube</td>
<td>1.76%</td>
<td>NNTP</td>
<td>2.73%</td>
<td>Skype</td>
<td>2.44%</td>
</tr>
<tr>
<td>9</td>
<td>Facebook</td>
<td>1.42%</td>
<td>Facebook</td>
<td>1.71%</td>
<td>NNTP</td>
<td>2.30%</td>
</tr>
<tr>
<td>10</td>
<td>Teredo</td>
<td>1.18%</td>
<td>Skype</td>
<td>1.42%</td>
<td>PPStream</td>
<td>1.77%</td>
</tr>
<tr>
<td></td>
<td><strong>Top 10</strong></td>
<td><strong>89.19%</strong></td>
<td><strong>Top 10</strong></td>
<td><strong>77.49%</strong></td>
<td><strong>Top 10</strong></td>
<td><strong>78.36%</strong></td>
</tr>
</tbody>
</table>

Source: Sandvine Network Demographics
Lessons Learned From the Past

- Specific events might significantly impact popularity of P2P protocols
  - Disconnection of popular services
    - Suprnova, mininova, ThePirateBay
  - Specific laws
    - 3-strikes, lawsuits

- This has always been a transient impact
Why to Study P2P (Old Version)

- P2P represents most of the Internet traffic
- Don’t you think there is a need for such a service
- And, in this case which techno will you use to reach millions of users without huge distribution costs
- Yes, but it is for illegal contents. P2P applications are evil
- Yes, but people should pay for the service and we need to keep control on it
- P2P
Why to Study P2P (New Version)

- BitTorrent is super fast to distribute contents
  - Start to be used by several big companies
    - e.g., Amazon, Spotify

- Twitter is using Murder to update Twitter servers (July 2010)
  - 75x faster
Murder

- Without Murder
- With Murder

Credit: Larry Gadea
Murder Performance

Credit: Larry Gadea
Outline

- Overview
- Content Replication
  - P2P performance
  - Parallel Download
  - Piece and Peer selection
Definitions

- Service capacity
  - Number of peers that can serve a content
  - It is 1 for client-server, constant with time

- Flash crowd of n
  - Simultaneous request of n peers (e.g., soccer match, availability of a patch, etc.)

- Piece (a.k.a. chunk, block)
  - A content is split in pieces
  - Each piece can be independently downloaded
Why P2P is so efficient?

- The service capacity grows exponentially with time.
- With a flash crowd of n peers, the mean download time is in $\log(n)$:
  - It is in $n$ for a client server model.
- The mean download time decreases in $1/($# of pieces$)$ when the # of pieces increases:
  - Do not take into account the overhead.
Intuition

- P2P
- Client-server
P2P vs. Client-Server

- **P2P**
  - Capacity of service $C(t) = O(e^t)$, where $t$ is time

- **Client-server**
  - Capacity of service $C(t) = 1$, where $t$ is time

- Time to serve a content: 10 minutes
P2P vs. Client-Server

- P2P
  - Download completion time $D(n) = O(\log(n))$, when $n$ is the number of peers

- Client-server
  - Download completion time $D(n) = n$, where $n$ is the number of client

- Time to serve a content: 10 minutes
Content Transfer Model

- Simple deterministic model [5]
  - Each peer serves only one peer at a time
  - The unit of transfer is the content
  - $n-1$ peers want the content
  - We assume $n=2^k$
  - $T$ is the time to complete an upload
    * $T=s/b$, $s$ content size, $b$ upload capacity
  - Peer selection strategy
    * Easy with global knowledge: Binary tree
Proof: Capacity

Capacity of service $C$

- $t=0$, $C=2^0$ peers
- $t=T$, $C=2^1$ peers
- $t=2T$, $C=2^2$ peers
- ...
- $t=iT$, $C=2^i$ peers

- $C=2^{t/T}$ peers
Proof: Finish Time

Seed

Finish time
- Seed has the content at $t=0$
- $2^0$ peers finish at $t=T$
- $2^1$ peers finish at $t=2T$
- ...$
- 2^{k-1}$ peers finish at $t=kT$
- We covered the $n$ peers
  - $1 + 2^0 + 2^1 + 2^2 + ... + 2^{k-1}$
  - $= 2^k$
  - $= n$
Proof: Finish Time

- Finish time
  - All peers have finished at $t = kT = T \cdot \log_2 n$

Seed

$t=0$
$t=T$
$t=2T$

time
Proof: Finish Time

Mean download time

\[ \bar{d} = \frac{1}{n} (1.0 + 2^0 \cdot T + 2^1 \cdot 2T + 2^2 \cdot 3T + \ldots + 2^{k-1} \cdot kT) \]

\[ \bar{d} = \frac{1}{n} \sum_{i=0}^{k-1} 2^i (i+1)T = \frac{T}{n} \sum_{i=0}^{k-1} 2^i (i+1) = \frac{T}{n} S_{k-1} \]

See refresher

\[ \bar{d} = \frac{T}{n} [(k-1) \cdot 2^k + 1] = T \left[ k - 1 + \frac{1}{n} \right] \]

\[ \bar{d} = T \left( \log_2 n + \frac{1-n}{n} \right) \]
Refresher

\[ S_{k-1} = \sum_{i=0}^{k-1} 2^i (i + 1) = 1.2^0 + 2.2^1 + 3.2^2 + \ldots + k.2^{k-1} \]

\[ S_{k-1} - 2S_{k-1} = 2^0 + 2^1 + 2^2 + \ldots + 2^{k-1} - k.2^k \]

\[ - S_{k-1} = \frac{2^k - 1}{2 - 1} - k.2^k = n - 1 - k.n \]

\[ S_{k-1} = n(k - 1) + 1 \]
Model Discussion

- Each peer has the same upload capacity
- No network bottleneck
- **Idealized peer selection strategy**
  - Each peer always knows to which peer $P$ to send the content at a given time
    - This peer $P$ does not have the content yet
    - This peer $P$ is not chosen by any other peer
  - Conflict resolution solved with global knowledge
  - No peer dynamics, i.e., arrival and departure

- **No piece selection strategy**
- **No advanced hypothesis:** reciprocation, parallel download, etc.
Piece Transfer Model

  - Each peer serves only one peer per time slot
  - The unit of transfer is a piece
  - n-1 peers want the content
  - We assume $n=2^k$
  - There are m pieces of the same size
  - We assume $m>k$
  - $S=s/(bm)=T/m$ is a time slot, s content size, b upload capacity
Results Summary

- Finished time
  - \( t = (T/m) \cdot \log_2 n + T \)
  - Decreases in \( 1/m \) compared to the content based model
  - Does not account for pieces overhead
Peer selection strategy (used in the model)

- We define $A_i$ as the set of peers that have piece $i$. We do not count in this set the seed
- Two strategies:
  - First strategy, when at least one peer has no piece
    - Peers send pieces to peers that have not yet obtained any piece.
Peer selection strategy

- Second strategy, when all peers have at least one piece
  - The set of peers $A_i$ with $n/2$ copies of $i$ replicate $i$ on the $n/2-1$ other peers. The $n/2-1$ other peers and the seed replicate pieces not present on the peers of $A_i$.
  - For instance, at $k+1$, $A_1$ replicate 1 on all the $A_i$, $i=2,\ldots,k$ and the $A_i$, $i=2,\ldots,k$ plus the seed replicate a piece on $A_1$
  - At each round one peer of $A_i$ is idle
First Peer Selection Strategy

- At $t=0$
  - Seed has all pieces
- At $t=S$
  - $|A_1|=2^0$
- At $t=2S$
  - $|A_1|=2^1$, $|A_2|=2^0$
- At $t=3S$
  - $|A_1|=2^2$, $|A_2|=2^1$, $|A_3|=2^0$
First Peer Selection Strategy

- At $t=jS$
  - $|A_i| = 2^{j-i}, \ i \leq j$

- This strategy ends when $j=k$
  - All $n-1 = 2^{k-1}$ leechers have a piece

$$\sum_{i=1}^{k} |A_i| = |A_1| + ... + |A_{k-1}| + |A_k|$$

$$= 2^{k-1} + ... + 2^1 + 2^0$$

$$= n - 1$$
Second Peer Selection Strategy

- We take as example $m=4$ and $k=3$
- In [5] they assume that the seed stops sending pieces when a copy of the content was served
  - Easier to model
  - Lower bound of the performance, because it uses less resources
Second Peer Selection Strategy

- We confirm that for $k=3$ all peers have a piece.

- $t=3S$
  - There are $2^{3/2}$ piece 1
  - There are $2^{3/2^2}$ piece 2
  - There are $2^{3/2^3}$ piece 3

- $t=4S$
  - All have piece 1
  - There are $2^{3/2}$ piece 2
  - There are $2^{3/2^2}$ piece 3
  - There are $2^{3/2^3}$ piece 4
Second Peer Selection Strategy

- **t=5S**
  - All have piece 1 and 2
  - There are $2^{3/2}$ piece 3
  - There are $2^3/2^2$ piece 4

- **t=6S**
  - All have piece 1, 2, and 3
  - There are $2^3/2^2$ piece 4

- **t=7S**
  - All have piece 1, 2, 3, and 4
Results

- At $t=kS$ each peer has a single piece
  - $|A_i| = 2^{k-i}$, $i \leq k$

- At slot $k+i$ for $i \leq m$
  - Each peer has pieces $1, \ldots, i$
  - $|A_{i+1}| = n/2$ peers have piece $i+1$ and replicate it on the $n/2-1$ other peers
    - The seed already has piece $i+1$
    - Hence, all have piece $i+1$
  - Each other peer replicates a piece on the peers in $A_{i+1}$
    - At the $m$ slot, the seed stops serving pieces
    - For all $j > i+1$, $|A_j| \rightarrow 2^*|A_j|$
Results

Finished time

- At each slot the number of copy of each piece is doubled
- When there are $n=2^k$ peers, a piece needs $k+1$ slots to be on all peers
  - We consider that the first slot for piece $x$ is when $x$ is sent by the seed to the first peer
- For $m$ pieces, $k+m$ slots are required to distribute all pieces on all peers
Results

- Finished time
  - All peers have finished at $t = (k+m)S$
  - $t = (k+m)S = T(k+m)/m = \frac{T}{m} \cdot \log_2 n + T$
  - Decreases in $1/m$ compared to the content based model
  - Does not account for pieces overhead
Results

Mean download time

- With the proposed strategy, at kS each peer has only one piece.
- As the number of pieces doubles at each slot, one needs \(k+m-1\) slots for half of the peers to have all the pieces.
  - At \(k\), 1 piece; at \(k+1\), 2 pieces; at \(k+m-1\), \(m\) pieces.
  - But at \(m\), the seed stops serving pieces, thus at \(k+m-1\) only half of the peers have \(m\) pieces, the rest have \(m-1\) pieces.
- The other half receives the last pieces at \(k+m\).
Results

- Mean download time

\[ \bar{d} = \frac{S}{2} [(k + m - 1) + (k + m)] \]

\[ \bar{d} = \frac{T}{m} \left( \log_2 n + m - \frac{1}{2} \right) \]

\[ \bar{d} = \frac{T}{m} \log_2 n + T \left( 1 - \frac{1}{2m} \right) \]
Model Discussion

- Each peer has the same upload capacity
- No network bottleneck
- **Idealized peer selection strategy**
  - Each peer always knows to which peer $P$ to send the content at a given time
  - Conflict resolution solved with global knowledge
  - No peer dynamics, i.e., arrival and departure
- **Idealized piece selection strategy**
  - Global knowledge
  - A peer is never blocked because it does not have the right piece to upload
- No advanced hypothesis: reciprocation, parallel download, etc.
- Read [5,6] for more sophisticated models
Model Discussion

The results obtained with this model hold for more complex models

- Stochastic

Lesson

- A simple model can give fundamental results
- Understand the assumptions and limitations
- No need for complexity if it is at the price of stronger or additional assumptions
Discussion of the Results

- P2P is very efficient when
  - There is always a peer to send data to
  - There is always a piece to send to this peer

- Peer and piece selection are at the core of an efficient P2P protocol
  - P2P efficiency can be from the idealized model to even worse than client-server

- How to select peers and pieces discussed in the following
Outline

- Overview
- Content Replication
  - P2P performance
  - Dynamic Parallel Download
  - Piece and Peer selection
Dynamic Parallel Download

- Parallel download
  - The principle to download from several server in parallel
- Dynamic parallel download
  - A parallel download with the following strategy
  - Strategy
    • Request first one piece from every server with the content
    • Each time a server has completed its upload of a piece, request a piece from this server that has not yet been requested from any other server
Dynamic Parallel Download: 4 pieces example
Performance Issues

- All servers must be busy sending pieces

- Two performance issues
  - Interblock idle time
    - Pipelining
  - Termination idle time
    - End game mode (Terminology introduced in BitTorrent)
Interblock Idle Time

- Time to receive a new request after sending the last byte of a piece
- Idle time = 1 RTT

Problem
- Server underutilized

Solution
- Pipelining
Pipelining

- Keep enough requests pending so that the server is never idle

- **1st solution**
  - Send request before the end of the current piece
  - RTT estimate
  - Piece transmission time > RTT
Pipelining

- 2\textsuperscript{nd} solution
  - Always have n pending requests
  - Still need RTT estimate
    - No need for accuracy
    - Overestimate does not harm
Termination Idle Time

- For dynamic parallel download from M servers
- \( P \) is the number of pieces not yet received
- When \( P < M \), \( M - P \) servers are idle
- Solution: end game mode
  - When \( P < M \) request pending blocks to all the idle servers
  - Several servers upload the same piece at the same time
    - The fastest win
  - Bandwidth waste: request + partial download
Termination Idle Time

- Without end game mode
  - Last pieces download speed unknown
- With end game mode
  - Last pieces download speed equal to at least the one of the fastest server
Experimental Evaluation

- Java client that implements dynamic parallel download
  - Does not implement pipelining
  - Implement a basic version of end game mode
- Connect to real mirror of public web servers in the Internet
- Study performed in 1999/2000
- For each figure is given the optimum transmission time
  - Ideal download time that would have been achieved in case there is neither interblock nor termination idle time (computed *a posteriori*)
No Shared Bottleneck

- The client connects to 4 mirror spread in the Internet: Japan, Portugal, Slovakia, Australia
  - High probability of disjoint paths, which implies no shared bottleneck
Results: No Shared Bottleneck

- Content size: 763KB
- # of pieces: 80
- Parallel: 4
- No shared bottleneck
- Parallel close to optimum

Credit: Rodriguez et al. [8]
Shared Bottleneck

- What happens when the bottleneck is the access link?
- The client is connected through a modem link (56kbit/s)
  - Connected to two slow servers (24kbit/s) and one fast server (56kbit/s)
- The fastest server is enough to saturate the access link
  - Dynamic parallel download will create TCP competition on a saturated link. What is the impact of that?
Results: Shared Bottleneck

- Content size
  - 256KB
- # of pieces
  - 20
- Parallel
  - 3
- Modem access line
  - Shared Bottleneck
- Close to the fastest server
  - Difference due to the interblock idle time

Credit: Rodriguez et al. [8]
Single Server vs. Multiple Servers Parallel Download

- Is it as efficient to perform parallel download from a single server as from multiple servers?
- The case of two mirrors: 1 fast and one slow
  - The client connects to a single mirror
  - The client connects to two mirrors in parallel
  - The client opens two TCP connections to the same mirror
Results: Single Server Parallel

- Content size: 256KB
- # of pieces: 20
- Parallel: 2
- No shared Bottleneck
- Close to the fastest server, but no need for server selection

Credit: Rodriguez et al. [8]
Properties

- Automatically adapt to the best servers and bottlenecks
  - No need for server selection
- No complex resource discovery
  - History based parallel access performs significantly worse
- Tradeoff
  - Piece request message overhead
  - Increase the number of TCP connections
Dynamic Parallel Download for P2P

- Dynamic parallel download
  - In the context of client-server
  - For a small number of parallel downloads

- P2P
  - Every peer is a client and a server
    - Parallel download and parallel upload
  - Large peer set

- Very different context
  - How to apply dynamic parallel download to P2P?
Dynamic Parallel Download for P2P

- A straightforward application to P2P
  - Every peer performs global dynamic parallel download to every other peer

Problems

- Not possible to maintain a large number of TCP connections per peer
- Why a peer should send data to another peer?
  - Not viable: free rider problem
Dynamic Parallel Download for P2P

Free rider problem

- A free rider is a peer that downloads without contributing anything
- To scale, each peer in a P2P system must act as a client and a server
- With global dynamic parallel download no incentive to do so
- We do not leave in an ideal word: selfish assumption
Dynamic Parallel Download for P2P

- Assume an **ideal word**
  - Each peer cooperate
  - Can we use dynamic parallel download?

- Studies on dynamic parallel **upload**
  - In P2P the content flow is from the initial seed toward leechers
  - *Easier to model dynamic parallel upload than dynamic parallel download*
  - Equivalent properties
Dynamic Parallel Download for P2P

- Dynamic parallel upload vs. download
  - Download
    - The client want to download as fast as possible
  - Upload
    - The source want to upload as fast as possible
  - Same problem
    - Find the fastest peer among a set without any knowledge
Dynamic Parallel Download for P2P

Outdegree
- Number of parallel uploads from a peer

Tradeoff
- Increasing the outdegree increases the number of peers served at the same time, but decreases the service rate to each peer \([5][9]\)
Dynamic Parallel Download for P2P

Results

- Biersack et al. [9] showed that an outdegree of 3 is optimal. An outdegree of 2 or 4 gives almost the same result.
  - Static scenario
  - Forest of tree
  - Assume uniform capacity of the peers
    - Upload=download
    - Same capacity for all peers
Dynamic Parallel Download for P2P

Results

- Yang et al. [5] showed that increasing the outdegree adversely impact the service capacity in case of static peers (no leave at the end of the download).
- But, in case of dynamic peers (peers leave the system with a given probability after completing the download) parallel upload can improve the service capacity.
  - Outdegree should be less than 10, marginal gain above 4
- Branching process model (stochastic)
Conclusion

- Even in an **ideal world** the outdegree should be **small**
  - Around 4

- This number might be **increased** in case of **high upload capacity**, but no study to understand the real impact
  - Probably makes sense in case of **heterogeneous peers**
    - The fast peer increases its number of parallel uploads to saturate its upload capacity
  - Probably dangerous in case of homogeneous peers
    - All peers increase their number of parallel uploads to saturate their upload capacity. But, in this case the global efficiency decreases as shown in [5,9]

- Used by BitTorrent mainline if max upload > 42 kB/s
  - uploads = int(math.sqrt(rate * .6))
Conclusion

- We are **not** in an ideal world
  - In case of free riders
    - The system is not viable
    - The analytical results do **not** hold
    - Dynamic Parallel upload cannot be used

- How to prevent free riders?
Outline

- Overview
- Content Replication
  - P2P performance
  - Parallel Download
  - Piece and Peer selection
Why a Peer and Piece Selection?

- Lets go back to the simple model

**Assumptions**
- *Always find a peer with an interesting piece to upload from* (Global knowledge)
  - Never idle or seeking for a peer
- *A peer never refuses to upload a piece*
  - No free riders

- If any of these assumptions is relaxed
  - No model for the system
  - No idea of its performance (at least worse)
  - No parallel download (selfish environment)
Why a Peer and Piece Selection?

- Additional assumptions
  - All peers have the same upload capacity
    - Always the best match
  - No network bottleneck
    - Still the best match

- This is **not** reality

- If best match relaxed
  - Performance decreases

- But, a good match is still possible in real life
Which Peer and Piece Selection?

- No specification except for BitTorrent
  - Always focus on content localization
- No similar problem in another field
- No general study of the problem
  - Always based on BitTorrent
Which Peer and Piece Selection?

- **Gnutella**
  - Designed for efficient content localization
  - No file splitting in the specification 0.6 [16]
  - Partial file transfer introduced in [17]
    - Allows peers with partial content to answer queries
  - Same heuristic for piece and peer selection
    - Select the first peer that answers the content request
    - Possibility of parallel download
  - Poor overall performance
    - No specific study of the file transfer efficiency
    - Mostly used for small contents (mp3)
Which Peer and Piece Selection?

- **Edonkey2000/Emule/Overnet**
  - Designed for efficient content localization
  - Only differ by their localization protocol
  - *File splitting* [13]
    - Rarest pieces first + other criteria with lesser priority
  - **Peer selection**
    - (Time spent in the priority queue) * (credit modifier based of upload and download rate)
    - Slow reactivity
    - Possibility of parallel download
  - **Average overall performance**
    - No specific study of the file transfer efficiency
Which Peer and Piece Selection?

- **BitTorrent**
  - Designed for *efficient file transfer*
  - *File splitting* [13]
    - Rarest pieces first
  - **Peer selection**
    - *Choke algorithm* based on short term peer upload speed estimation
    - Fast adaptation
    - Use of parallel download
  - *Good overall performance*
    - Several specific studies
Which Peer and Piece Selection?

- **Common mistakes** made about BitTorrent (BT)
  - With BT contents are hard to find
    - Right, BT is a file transfer protocol not a localization protocol
    - Does it make sense to say that with HTTP contents are hard to find?
  - With BT a torrent die when there is no more seed
    - Right, BT is a file transfer protocol, not an infrastructure that manage persistency
    - Does it make sense to say that HTTP does not guarantee that your web server is always up?

- **BT is a P2P file transfer protocol, nothing more**
Which Peer and Piece Selection?

- In the following, general discussion, but based on the experience gathered with BitTorrent
  - BitTorrent is the state of the art
  - Might be improved, but need a deep understanding
Which Peer and Piece Selection?

- **Peer selection task**
  - Always find a peer to upload from
  - Prevent free riders
  - Converge to the *best* upload-download match

- **Piece selection task**
  - Piece diversity is called *entropy*
  - With ideal entropy, each peer always has an interesting piece for any other peer
Which Peer and Piece Selection?

- **Peer selection** must **not** have a piece constraint
  - Ideal entropy is the target
  - Peer selection should be **based on capacity only**, **not** on piece availability
Selection Order

- Performs piece then peer selection
  - Puts a constraint on the peer to select
  - The selected peer is unlikely to be a good match
  - Depending on the piece selection, may create hot spots
    - Rare piece selection, with an initial seed with all the pieces
  - Focus on the piece then on the capacity of service
I want coolContent.xvid
Selection Order

- Performs peer then piece selection
  - Select first the **best match peer**
  - Then apply the *piece selection on that peer*
  - Focus on the **capacity** of service first
Peer then Piece selection

- Which is the best peer (among my neighbors)?
- What do you have?

coolContent.xvid

P1 P2 P3
Selection Order

- Performs peer then piece selection
- Peer and piece selections are interlinked
  - To find the best match you may need pieces to download to test if it is the best match
  - No sense to select a peer with no interesting pieces
Selection Order

- Peer selection first is a better strategy
  - Important to maximize the capacity of service
  - No study on this order

- No general reflection on the role of peer and piece selection
  - Results are given as reasonable rules
  - Experience on existing P2P protocols
Piece Selection

- **Random piece selection**
  - Each peer selects at random a piece to download

- **Global rarest first**
  - Each peer maintains globally the number of copies of each piece
  - Select the **globally rarest piece** to download
  - Require global knowledge
Piece Selection

- Local rarest first (LRF)
  - Approximation of global rarest first
  - Each peer maintains the number of copies in its peer set of each piece
  - Select the **locally rarest piece** to download

- When peer selection is performed first, rarest first piece selection is **applied on the pieces available on the selected peer**
Piece Selection: LRF

- Local rarest first algorithm
  - Choose the pieces that are locally rarest

coolContent.xvid

P1  P2  P3
Piece Selection Properties

- Random piece selection performs poorly [33]
  - Last pieces problem
  - Poor entropy, i.e., high constraint on peer selection
- Global rarest first [33]
  - Good entropy
- Local rarest first [18, 33, 34]
  - Good entropy with a large peer set
  - Care should be taken to the graph construction (random graph)
  - Inspire yourself from BitTorrent
No serious alternative to the BitTorrent Choke algorithm

Choke algorithm [18]
- Different algorithm in leecher and seed state
- Peer selection performed in the peer set
- **Choke/unchoke**
  - A **choke**s B if A decides to do **not upload** to B
  - A **unchoke**s B if A decides to **upload** to B
- **Interested/not interested**
  - A is interested in B if B has at least one piece that A does not have
  - A is not interested in B if B has a subset of the pieces A already has
Peer X is interested in peer Y if peer Y has at least 1 piece that peer X does not have.
Choke Algorithm: LS

- Leecher state (high level description)
  - Every 10 seconds the peers are ordered according to their download rate to the local peer
    - Tit for tat
  - The 3 fastest and interested peers are unchoked (Regular Unchocke RU)
  - Every 30 seconds, one additional interested peer is unchoked at random
    - Called the Optimistic Unchoke (OU)
    - If this peer is one of the three fastest, perform another OU
Choke Algorithm: LS

- Leecher state (high level description)
  - No more than 4 interested peers are unchoked at the same time
Choke Algorithm: LS

Time (10 seconds slots)

Peer ID

OU

RU
Real Torrent, LS

- 1 seed, 26 leechers
  - At torrent startup
- 350 MB
- Few peers with a lot of RU
- Uniform OU
Choke Algorithm: SS

- Seed state, version FU (high level description)
  - FU: Favor Upload
  - Oldest version, but still largely used today
  - Same as in leecher state, but peers are ordered according to their upload rate from the local peer
  - In seed state, there is no download
Choke Algorithm: SS

- Seed state, version RR (high level description)
  - RR: Round Robin
  - Appeared after the version FU, but not largely used today
  - Every 10 seconds the interested peers are ordered according to the time they were last unchoked
    - Most recently unchocked peers first
  - For two consecutive periods of 10 seconds, the 3 first peers are unchoked and an additional 4th interested peer is unchoked at random
  - For the third period of 10 seconds, the 4 first peers are unchoked
Choke Algorithm: SS

Peer ID (oldest first)

Time (10 seconds slots)

10s

60 seconds

134
Real Torrent, SS

- 1 seed, 26 leechers
  - At torrent startup
- 350 MB
- Random SRU
Choke Algorithm Properties

- **Leecher state**
  - Robust to free riders
    - Only contributing peers get a good service from a leecher
  - Leechers unchoked based on download evaluation
    - Selects the fastest interested peers
- **Optimistic unchoke**
  - Capacity discovery mechanism
- **Global properties (see later)**
  - Clusters peers according to their upload speed
  - Ensures efficient sharing incentive
  - Achieves high upload utilization
Choke Algorithm Properties

- **Seed state**
  - Algorithm (Favour Upload) FU still implemented in every client except
    - mainline 4.x.y, Ctorrent
  - **Algorithm FU**
    - **Not** robust to free riders
      - The fastest peers get the service even if they do not contribute anything
    - **Bad piece diversity**
      - A single free rider can get most of the pieces
Choke Algorithm Properties

- **Seed state**
  - Algorithm RR robust to free riders
    - Every peer gets the same service time
  - Increases the piece diversity

- **Why FU more popular than RR**
  - RR is not well known (deployed experimentally on mainline 4.x.y)
  - FU is more efficient in the present context
    - Few contributing peers with a large capacity
Peer and Piece Selection

- Local rarest first piece selection [34]
  - Close to ideal entropy
  - Simple and efficient
  - No alternative today except network coding

- Choke algorithm (seed algorithm RR) [34,35]
  - Achieves high upload utilization
  - Clusters peers with similar upload capacity
  - Ensures effective sharing incentive
    - Robust to free riders
  - P2P fair
Peer and Piece Selection

- Detailed description during BitTorrent presentation
  - Many important implementation details

- Extensive evaluation during BitTorrent presentation
  - Rarest first entropy
  - Choke algorithm fairness and efficiency