

Peer-To-Peer Applications

Antonio Carzaniga

Faculty of Informatics
Università della Svizzera italiana

March 16, 2020

- Transferring big files
 - ▶ client-server vs. peer-to-peer
- BitTorrent
- Peer-to-peer search
- Miscellaneous

- How long does it take to transfer a *big* file?

- How long does it take to transfer a *big* file?

Example:

size = 6Gb (e.g., a movie file)

rate = 2Mb/s

T = ?

- How long does it take to transfer a *big* file?

Example:

size = 6Gb (e.g., a movie file)

rate = 2Mb/s

$T = 3000s = 50min$

- How long does it take to transfer a *big* file?

Example:

size = 6Gb (e.g., a movie file)

rate = 2Mb/s

$T = 3000s = 50min$

In general:

$$T = \frac{\textit{size}}{\textit{rate}}$$

(we ignore the $O(1)$ propagation delay, which is marginal for large files)

- How long does it take to transfer a *big* file?

Example:

size = 6Gb (e.g., a movie file)

rate = 2Mb/s

$T = 3000s = 50min$

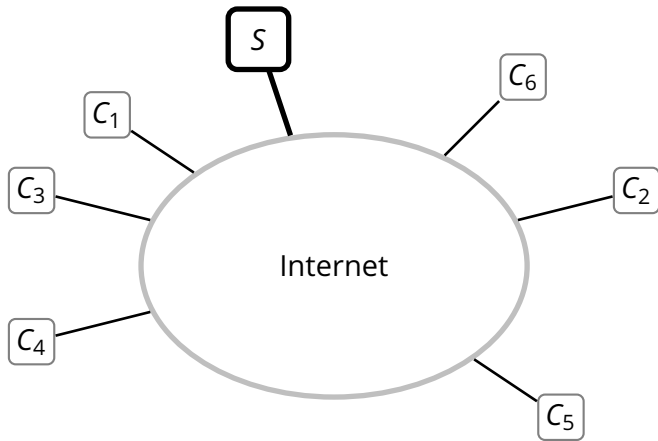
In general:

$$T = \frac{\textit{size}}{\textit{rate}}$$

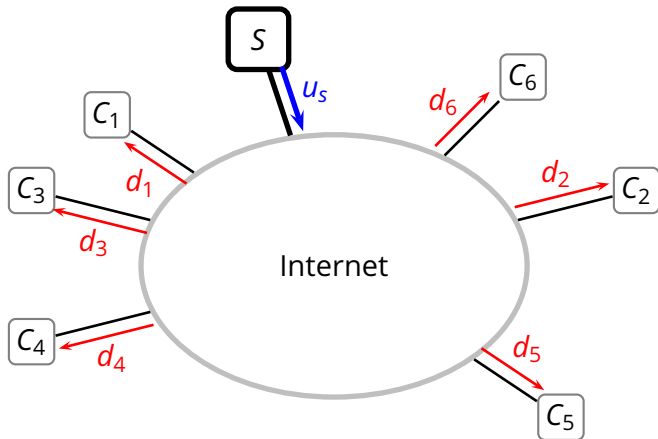
(we ignore the $O(1)$ propagation delay, which is marginal for large files)

- How long does it take to transfer a *big* and **very popular** file?
 - ▶ N clients want the file

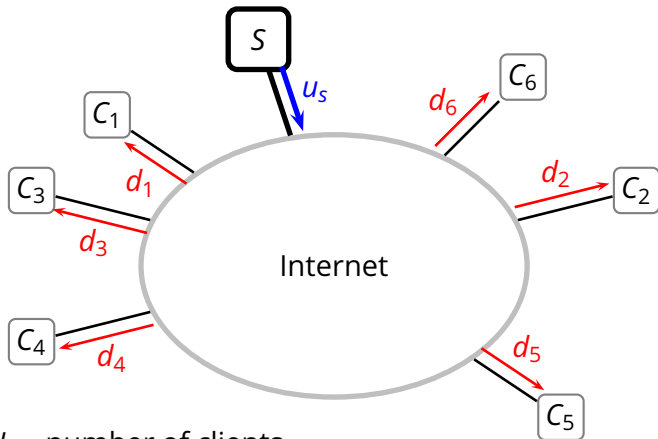
Transferring Big and Popular Files



Transferring Big and Popular Files

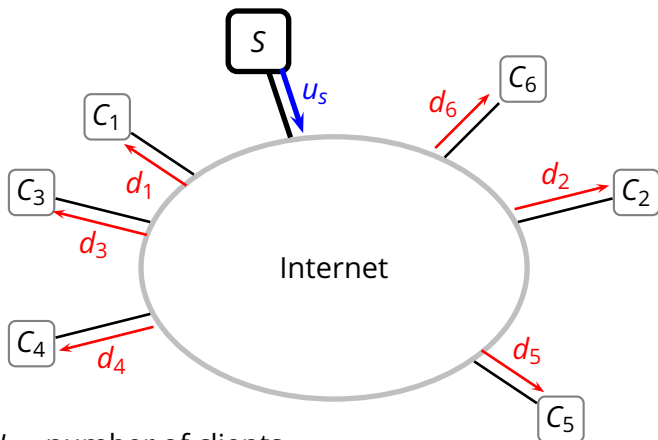


Transferring Big and Popular Files



Let F = file size, N = number of clients

Transferring Big and Popular Files



Let F = file size, N = number of clients

$$T_{CS} \geq \max\left(\frac{NF}{u_s}, \frac{F}{d_{min}}\right)$$

Exploiting Peer-to-Peer Connections

Exploiting Peer-to-Peer Connections

1. Split the file into *blocks*

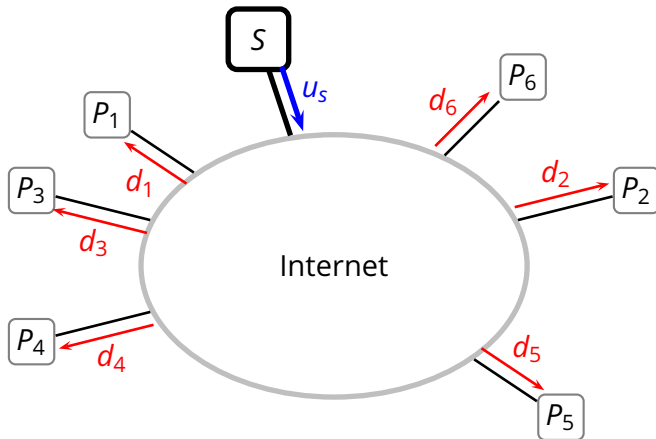
Exploiting Peer-to-Peer Connections

1. Split the file into *blocks*
2. The server sends different blocks to different clients

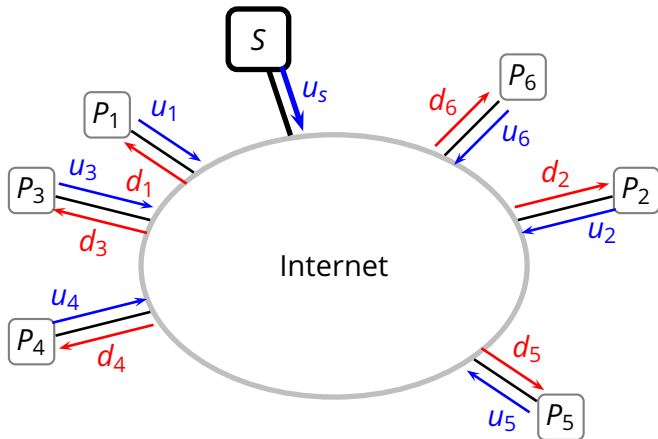
Exploiting Peer-to-Peer Connections

1. Split the file into *blocks*
2. The server sends different blocks to different clients
3. The clients exchange blocks using “peer-to-peer” connections

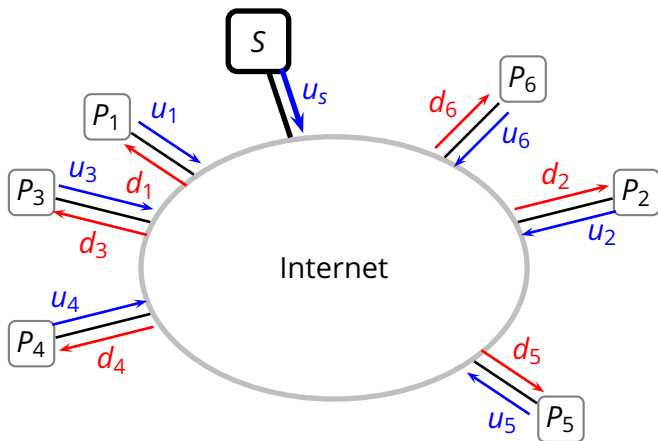
Transfer Time with P2P Connections



Transfer Time with P2P Connections



Transfer Time with P2P Connections



$$T_{P2P} \geq \max \left(\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i} \right)$$

Transferring Big and Popular Files

- Transfer time is *at least*

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i}\right)$$

Transferring Big and Popular Files

- Transfer time is *at least*

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i}\right)$$

- Assuming $u_1 = u_2 = \dots = u_N = u_p$

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{F}{u_s/N + u_p}\right)$$

Transferring Big and Popular Files

- Transfer time is *at least*

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i}\right)$$

- Assuming $u_1 = u_2 = \dots = u_N = u_p$

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{F}{u_s/N + u_p}\right)$$

- And for large peer groups ($N \gg 1$)

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{F}{u_p}\right)$$

Transferring Big and Popular Files

- Transfer time is *at least*

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i}\right)$$

- Assuming $u_1 = u_2 = \dots = u_N = u_p$

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{F}{u_s/N + u_p}\right)$$

- And for large peer groups ($N \gg 1$)

$$T_{P2P} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{F}{u_p}\right)$$

The transfer time does not depend on the number of receivers!

BitTorrent: Tracker and Startup

BitTorrent: Tracker and Startup

- A *tracker* keeps track of which peers participate in the “torrent”

BitTorrent: Tracker and Startup

- A *tracker* keeps track of which peers participate in the “torrent”
 - ▶ at startup, Alice requests a list of peers from the tracker
 - ▶ then, Alice tries to establish direct connections with her “neighbor peers”
 - ▶ periodically, Alice tells the tracker that she is still participating in the torrent

BitTorrent: Tracker and Startup

- A **tracker** keeps track of which peers participate in the “torrent”
 - ▶ at startup, Alice requests a list of peers from the tracker
 - ▶ then, Alice tries to establish direct connections with her “neighbor peers”
 - ▶ periodically, Alice tells the tracker that she is still participating in the torrent
- The torrent (one or more files) is split into **equal-size chunks**
 - ▶ peers accumulate chunks and keep track of the chunks they have
 - ▶ it might be that no single peer has all the chunks, as long as all the chunks are available from *some* peer

BitTorrent: Exchanging Chunks

BitTorrent: Exchanging Chunks

- Neighboring peers exchange their lists of chunks and eventually exchange chunks

BitTorrent: Exchanging Chunks

- Neighboring peers exchange their lists of chunks and eventually exchange chunks
 - ▶ periodically, Alice requests the list of chunks of her peers

BitTorrent: Exchanging Chunks

- Neighboring peers exchange their lists of chunks and eventually exchange chunks
 - ▶ periodically, Alice requests the list of chunks of her peers
 - ▶ Alice figures who has what chunks, and therefore requests some chunks from her neighbors
 - ▶ Alice requests the *rarest chunk first* (why?)

BitTorrent: Exchanging Chunks

- Neighboring peers exchange their lists of chunks and eventually exchange chunks
 - ▶ periodically, Alice requests the list of chunks of her peers
 - ▶ Alice figures who has what chunks, and therefore requests some chunks from her neighbors
 - ▶ Alice requests the *rarest chunk first* (why?)
 - ▶ Alice also receives requests from her neighbors

BitTorrent: Exchanging Chunks

- Neighboring peers exchange their lists of chunks and eventually exchange chunks
 - ▶ periodically, Alice requests the list of chunks of her peers
 - ▶ Alice figures who has what chunks, and therefore requests some chunks from her neighbors
 - ▶ Alice requests the *rarest chunk first* (why?)
 - ▶ Alice also receives requests from her neighbors
 - ▶ Alice gives priority to neighbors that share the most (highest rate): she sends her chunks to the top four (why?)

BitTorrent: Exchanging Chunks

- Neighboring peers exchange their lists of chunks and eventually exchange chunks
 - ▶ periodically, Alice requests the list of chunks of her peers
 - ▶ Alice figures who has what chunks, and therefore requests some chunks from her neighbors
 - ▶ Alice requests the *rarest chunk first* (why?)
 - ▶ Alice also receives requests from her neighbors
 - ▶ Alice gives priority to neighbors that share the most (highest rate): she sends her chunks to the top four (why?)
 - ▶ periodically, Alice also selects another trading partner at random (why?)

- How do you find files in a file-sharing network?

- How do you find files in a file-sharing network?
- Centralized index (i.e., client-server)
 - ▶ typically maps objects (e.g., files, nicknames) to IP addresses
 - ▶ not too good: performance bottleneck, single point of failure, etc.

- How do you find files in a file-sharing network?
- Centralized index (i.e., client-server)
 - ▶ typically maps objects (e.g., files, nicknames) to IP addresses
 - ▶ not too good: performance bottleneck, single point of failure, etc.
- Distributed peer-to-peer search with *query flooding*

- How do you find files in a file-sharing network?
- Centralized index (i.e., client-server)
 - ▶ typically maps objects (e.g., files, nicknames) to IP addresses
 - ▶ not too good: performance bottleneck, single point of failure, etc.
- Distributed peer-to-peer search with *query flooding*
 - ▶ not too good (why?)

- How do you find files in a file-sharing network?
- Centralized index (i.e., client-server)
 - ▶ typically maps objects (e.g., files, nicknames) to IP addresses
 - ▶ not too good: performance bottleneck, single point of failure, etc.
- Distributed peer-to-peer search with *query flooding*
 - ▶ not too good (why?)
 - ▶ many variants: limited scope, probabilistic, hierarchical with super-peers, etc.

- How do you find files in a file-sharing network?
- Centralized index (i.e., client-server)
 - ▶ typically maps objects (e.g., files, nicknames) to IP addresses
 - ▶ not too good: performance bottleneck, single point of failure, etc.
- Distributed peer-to-peer search with ***query flooding***
 - ▶ not too good (why?)
 - ▶ many variants: limited scope, probabilistic, hierarchical with super-peers, etc.
- Distributed peer-to-peer search with ***structured indexes***
 - ▶ a.k.a., ***distributed hash tables (DHTs)***

- How do you find files in a file-sharing network?
- Centralized index (i.e., client-server)
 - ▶ typically maps objects (e.g., files, nicknames) to IP addresses
 - ▶ not too good: performance bottleneck, single point of failure, etc.
- Distributed peer-to-peer search with ***query flooding***
 - ▶ not too good (why?)
 - ▶ many variants: limited scope, probabilistic, hierarchical with super-peers, etc.
- Distributed peer-to-peer search with ***structured indexes***
 - ▶ a.k.a., ***distributed hash tables (DHTs)***
 - ▶ many variants, lots of interesting theoretical and practical developments

- How does Skype work?

- How does Skype work? ...or what Skype does not want you to know

- How does Skype work? ... or what Skype does not want you to know
- **Search:** how does Alice find Bob?
 - ▶ peer-to-peer index

- How does Skype work? ... or what Skype does not want you to know
- **Search:** how does Alice find Bob?
 - ▶ peer-to-peer index
- **Connections:** how does Alice connect to Bob?

- How does Skype work? ... or what Skype does not want you to know
- **Search:** how does Alice find Bob?
 - ▶ peer-to-peer index
- **Connections:** how does Alice connect to Bob?
 - ▶ direct connections when possible

- How does Skype work? ... or what Skype does not want you to know
- **Search:** how does Alice find Bob?
 - ▶ peer-to-peer index
- **Connections:** how does Alice connect to Bob?
 - ▶ direct connections when possible
 - ▶ indirect connections through a relay "super-peer"

- How does Skype work? ... or what Skype does not want you to know
- **Search:** how does Alice find Bob?
 - ▶ peer-to-peer index
- **Connections:** how does Alice connect to Bob?
 - ▶ direct connections when possible
 - ▶ indirect connections through a relay "super-peer"
- And much more: chat, audio/video codecs, multi-party communication, etc.