Peer-to-Peer Applications From BitTorrent to Privacy

Arnaud Legout

INRIA, Sophia Antipolis, France
Projet Planète

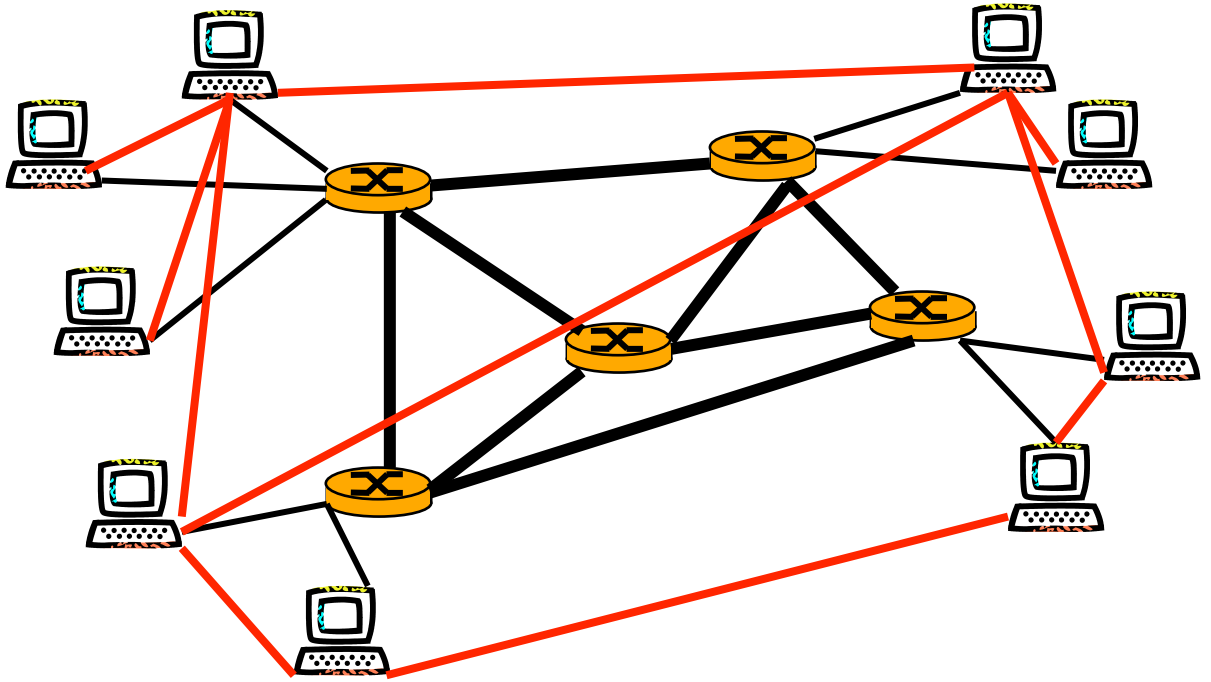
Email: arnaud.legout@inria.fr

The original set of slides is much complete, this small set is a selection made by C. Pham, UPPA.

version 2
January 2012

Definition: Overlay

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



Definition: Overlay

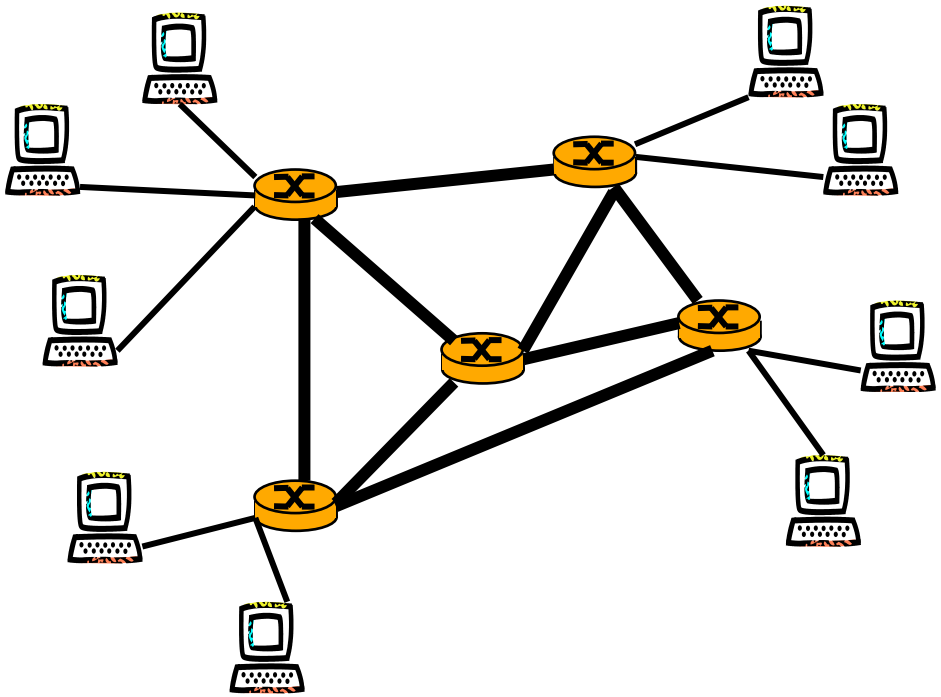
- Formed by communicating among themselves
  - Dedicated machines
  - End-users
- Types of overlay
  - General purpose overlay (application-layer multicast)
  - Application specific overlay (CDN)
- Overlay construction
  - Network topology
  - Network metrics (delay, bandwidth, etc.)

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

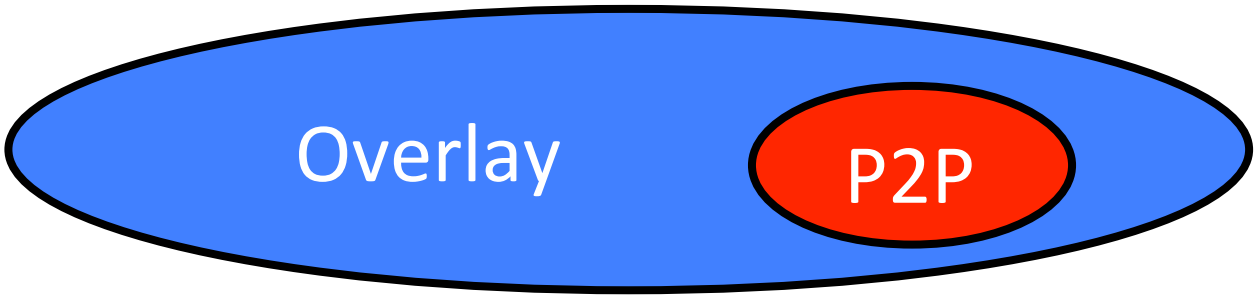


Definition: Peer

- 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

- 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



Example: Web

- 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
    - End-to-end argument [7] (1984) undoubtedly visionary
      - Still alive (01/2006): http://lwn.net/Articles/169961/
- P2P applications are a true revolution
  - Aside TCP/IP and the Web

New P2P applications

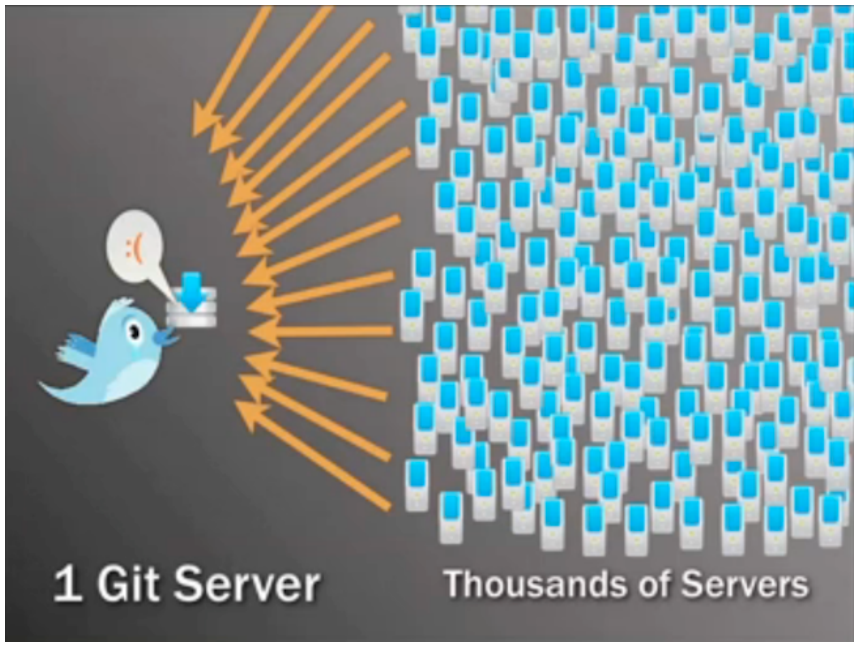
- P2P applications capitalize on any resource from anybody
  - P2P applications can share CPU, bandwidth and storage
    - seti@home (not P2P, but distributed)
    - BitTorrent, Emule, Gnutella
    - Skype, Google talk
    - Publius

Why to Study P2P (New Version)

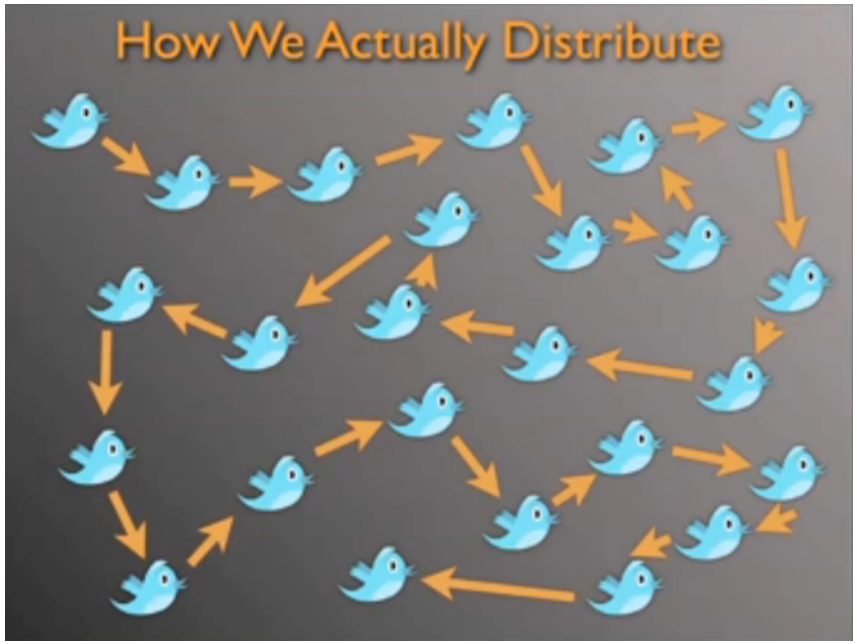
- BiTorrent is super fast to distribute contents
  - Start to be used by several big companies
- Twitter is using Murder to update Twitter servers (July 2010)
  - 75x faster
  - http://engineering.twitter.com/2010/07/murder-fast-datacenter-code-deploys.html

Murder

- Without Murder

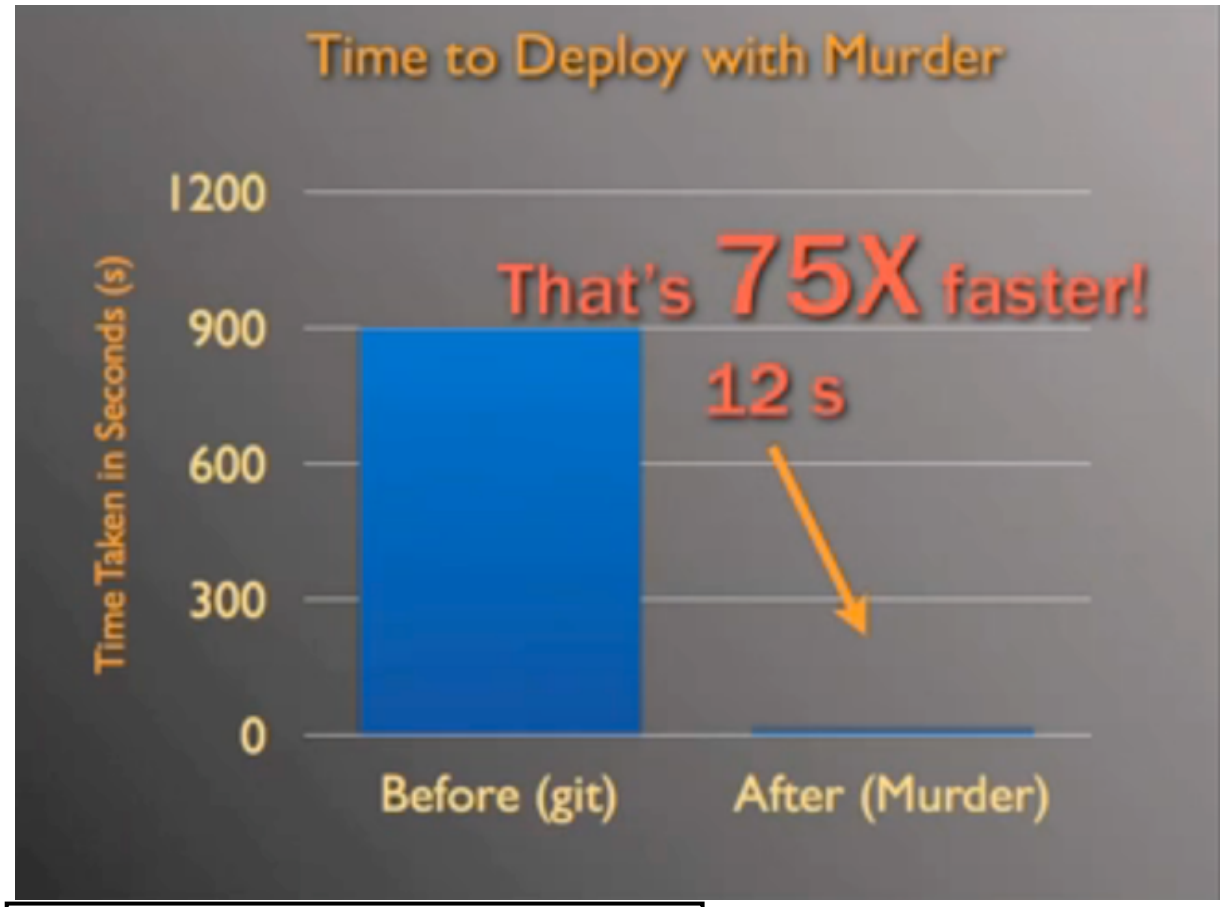


- With Murder



Credit: Larry Gadea

Murder Performance



Credit: Larry Gadea

Definitions

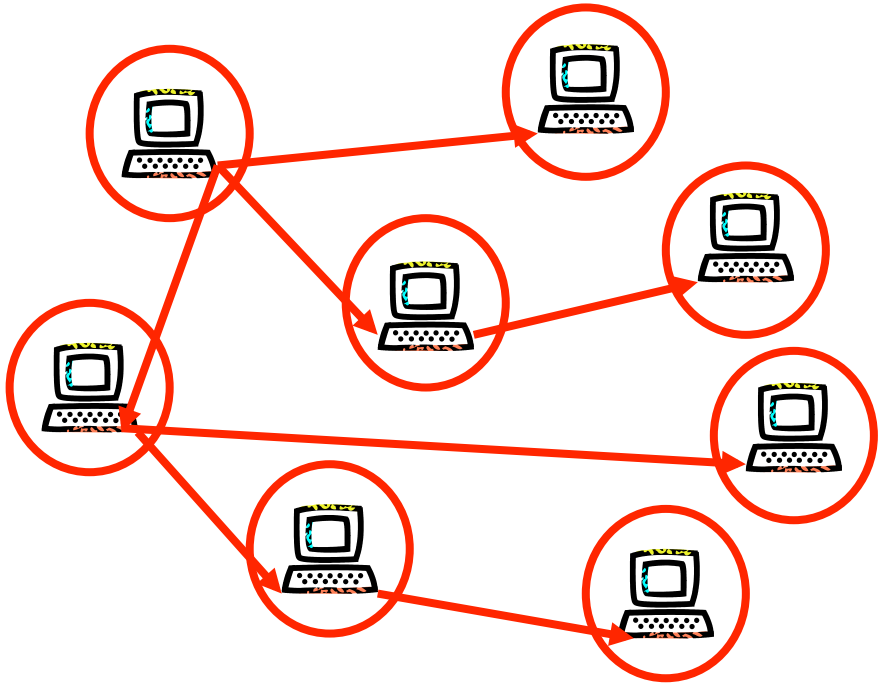
- Service capacity
  - Number of peers that can serve a content
  - It is 1 for client-server, constant with time
- Flash crowd of n
  - Simultaneous request of n peers (e.g., soccer match, availability of a patch, etc.)
- Piece (a.k.a. chunk, block)
  - A content is split in pieces
  - Each piece can be independently downloaded

Why P2P is so efficient?

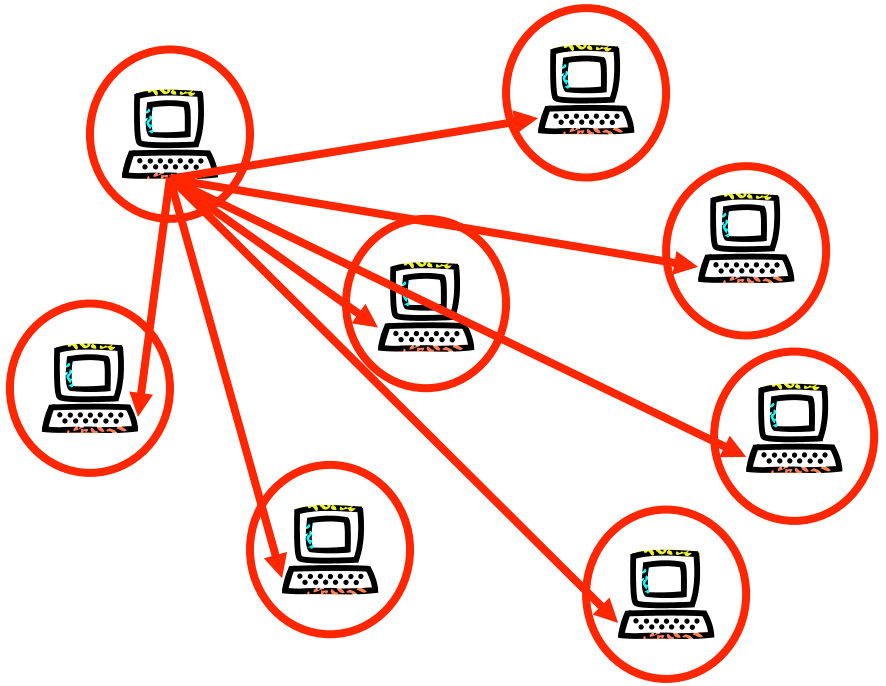
- The service capacity grows exponentially with time
- With a flash crowd of n peers, the mean download time is in *log(n)*
  - It is in *n* for a client server model
- The mean download time decreases in 1/(# of pieces) when the # of pieces increases
  - Do not take into account the overhead

Intuition

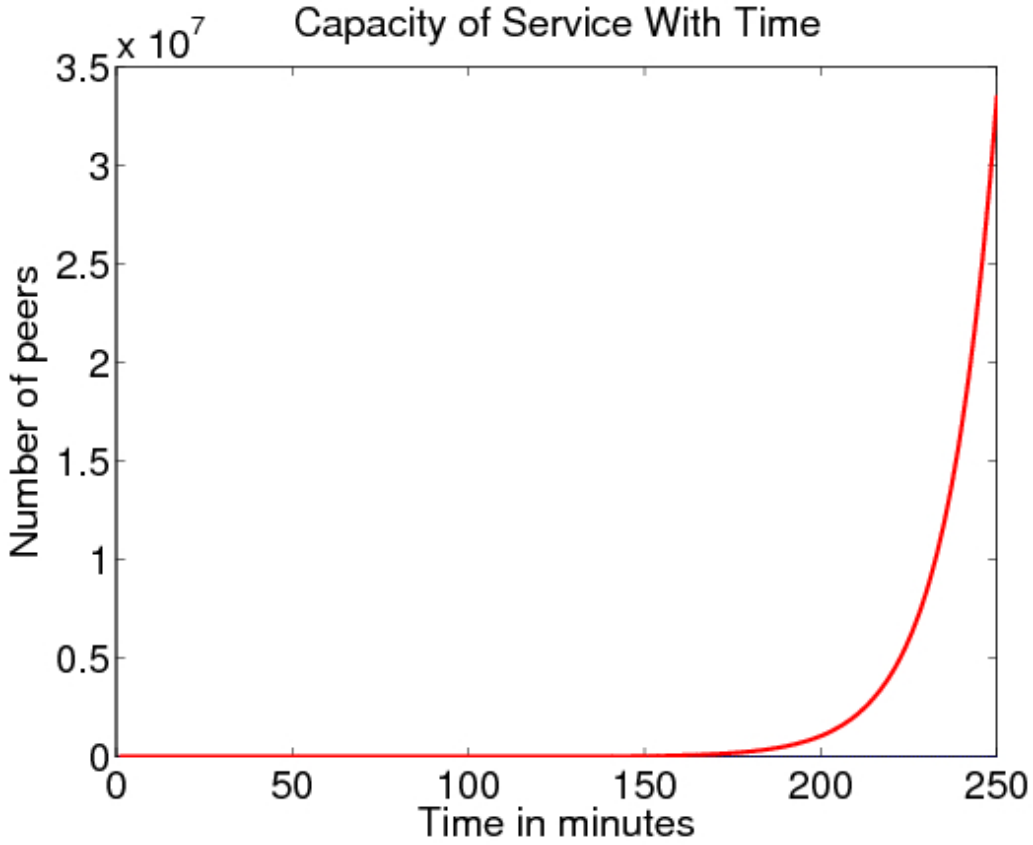
- P2P



- Client-server

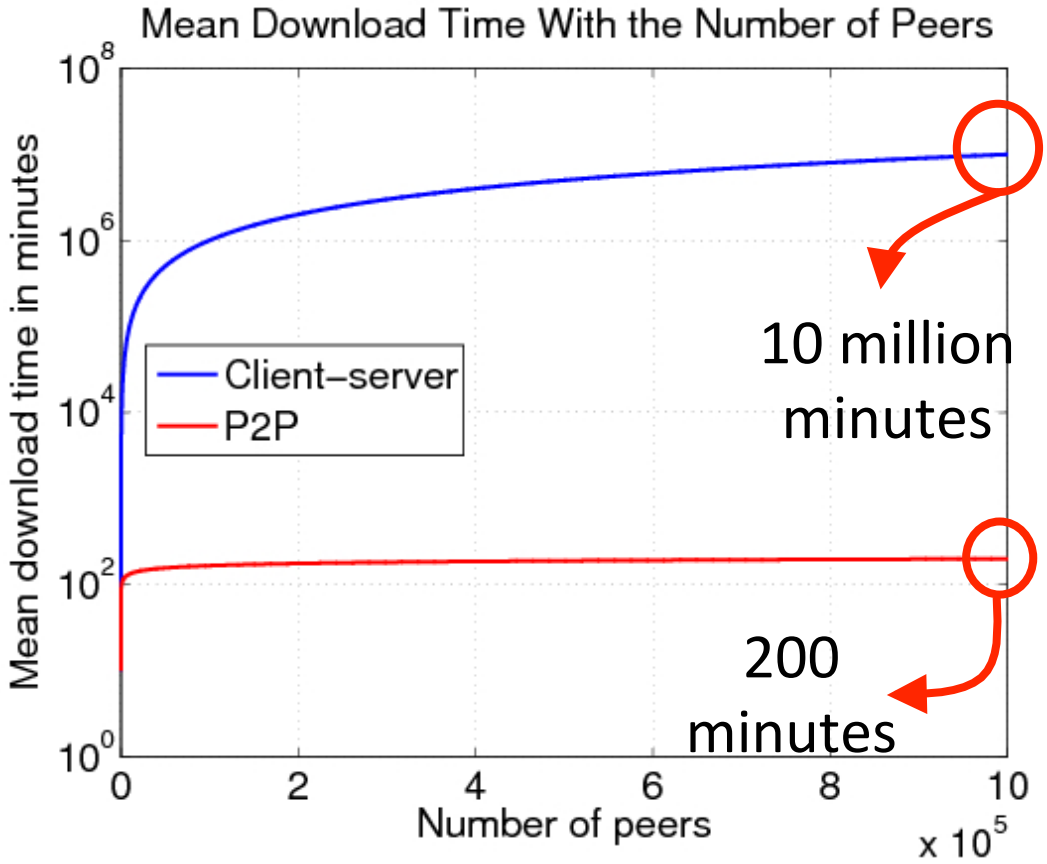


P2P vs. Client-Server



- 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

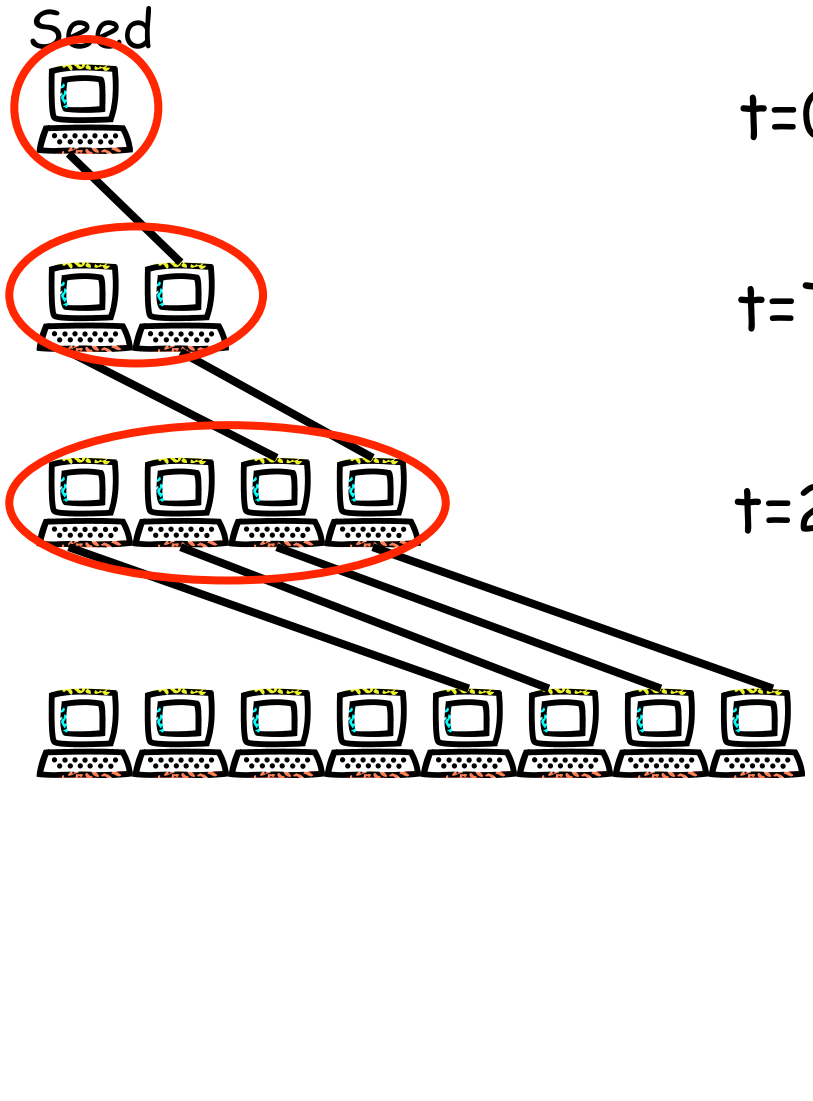


- P2P
  - Download completion time D(n)=O(log(n)), when n is the number of peers
- Client-server
  - Download completion time D(n)=n, where n is the number of client
- Time to serve a content: 10 minutes

Content Transfer Model

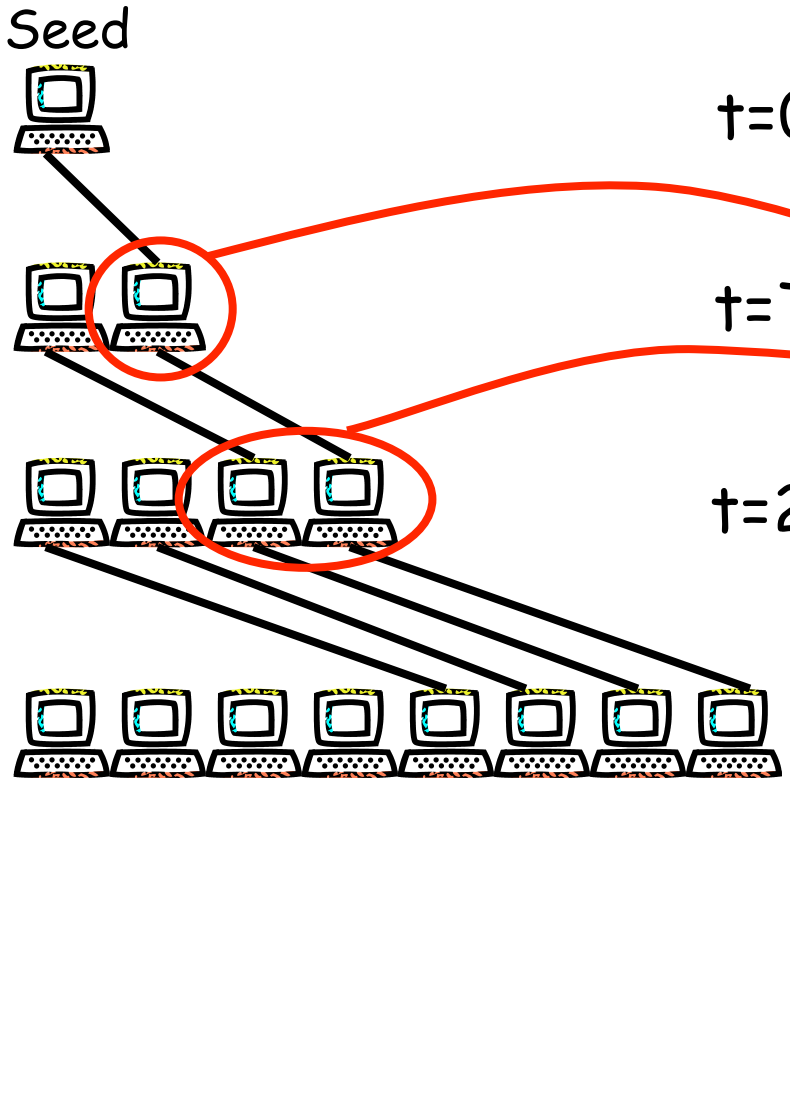
- Simple deterministic model [5] (to read)
  - Each peer serves only one peer at a time
  - The unit of transfer is the content
  - n-1 peers want the content
  - We assume $n=2^k$
  - T is the time to complete an upload
    - T=s/b, s content size, b upload capacity
  - Peer selection strategy
    -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

Proof: Capacity



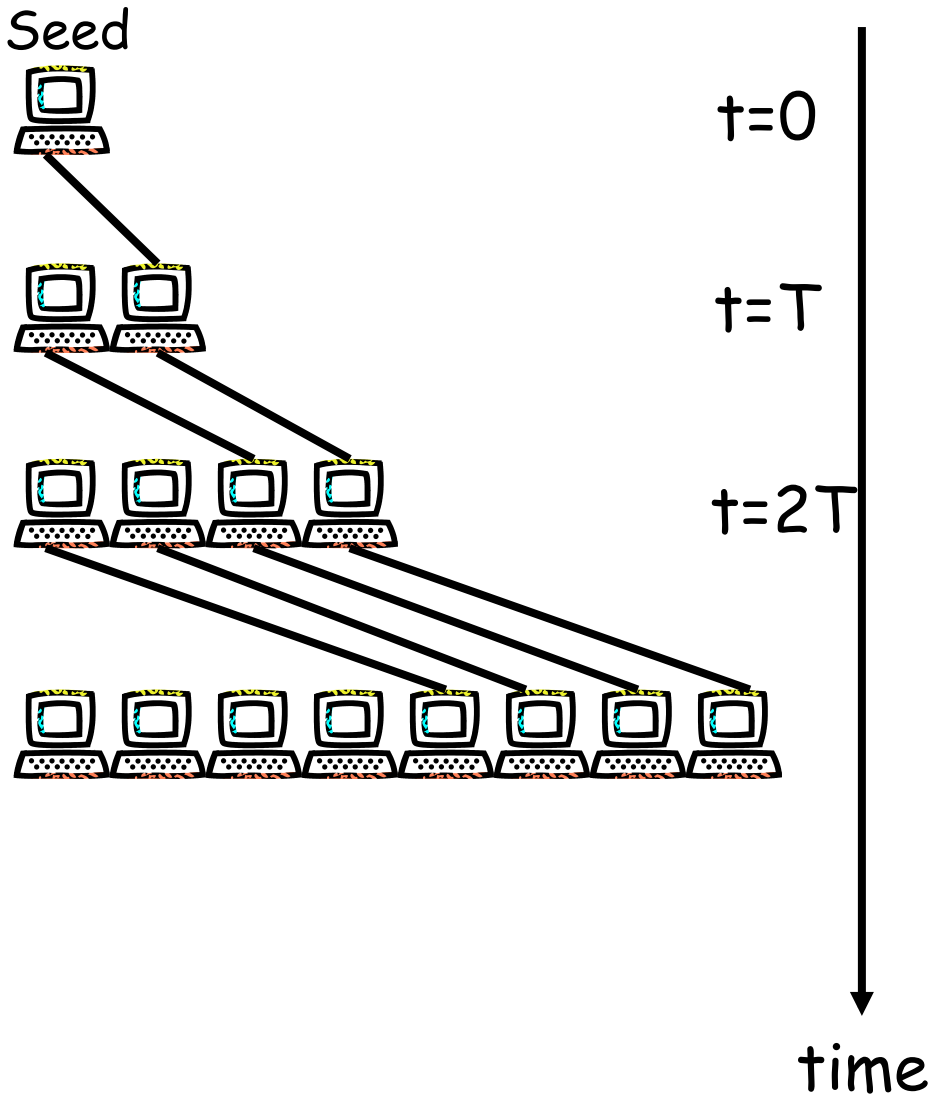
- Capacity of service C
  - t=0, $C=2^0$ peers
  - t=T $C=2^1$ peers
  - t=2T $C=2^2$ peers
  - …
  - t=iT $C=2^i$ peers
  - $C=2^{t/T}$ peers

Proof: Finish Time



- 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

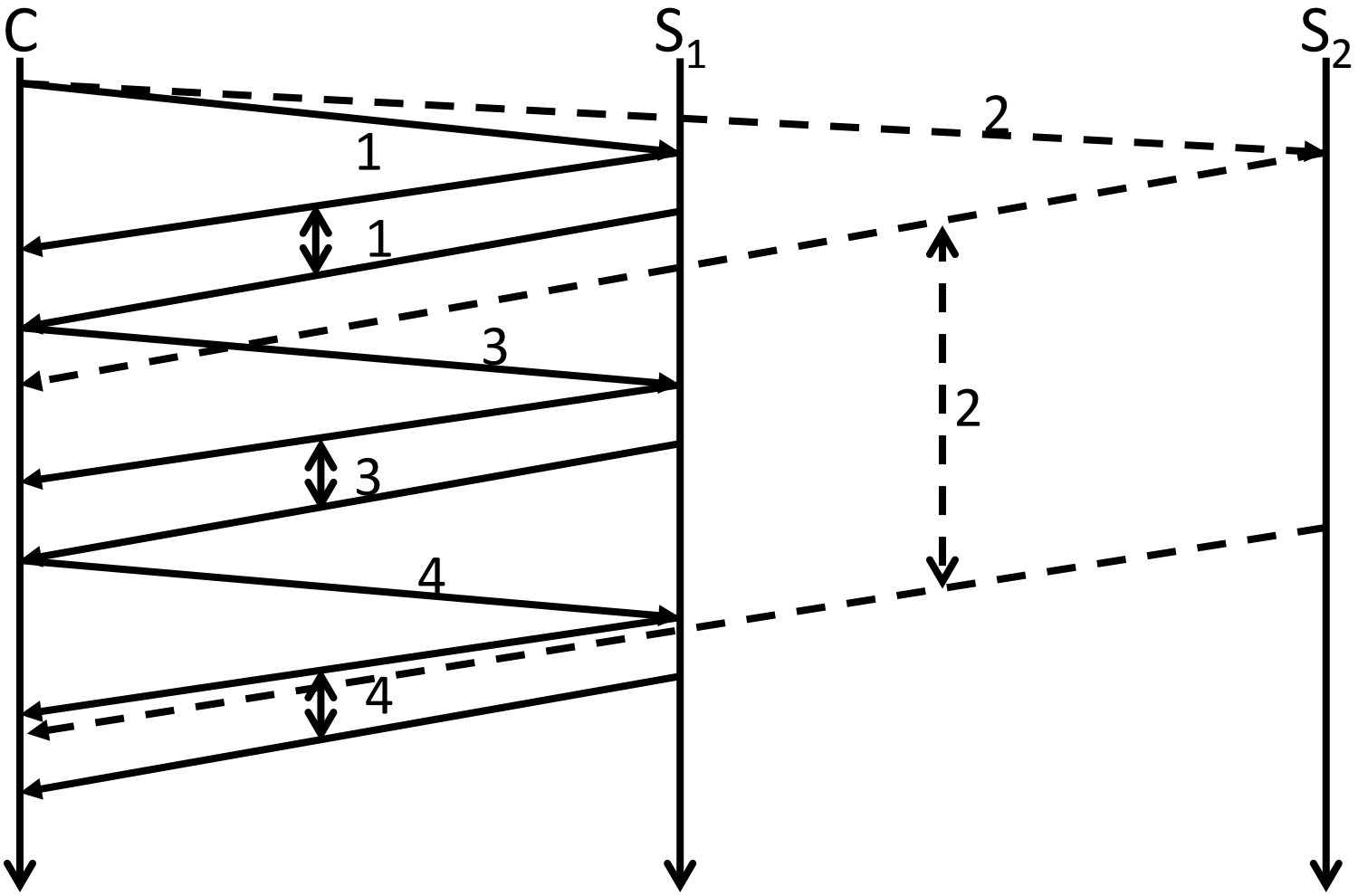


- Finish time
  - All peers have finished at $t=kT=T.\log_2 n$

Dynamic Parallel Download

- Introduced by Rodriguez et al. [8] (2000) in the context of web cache
- 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

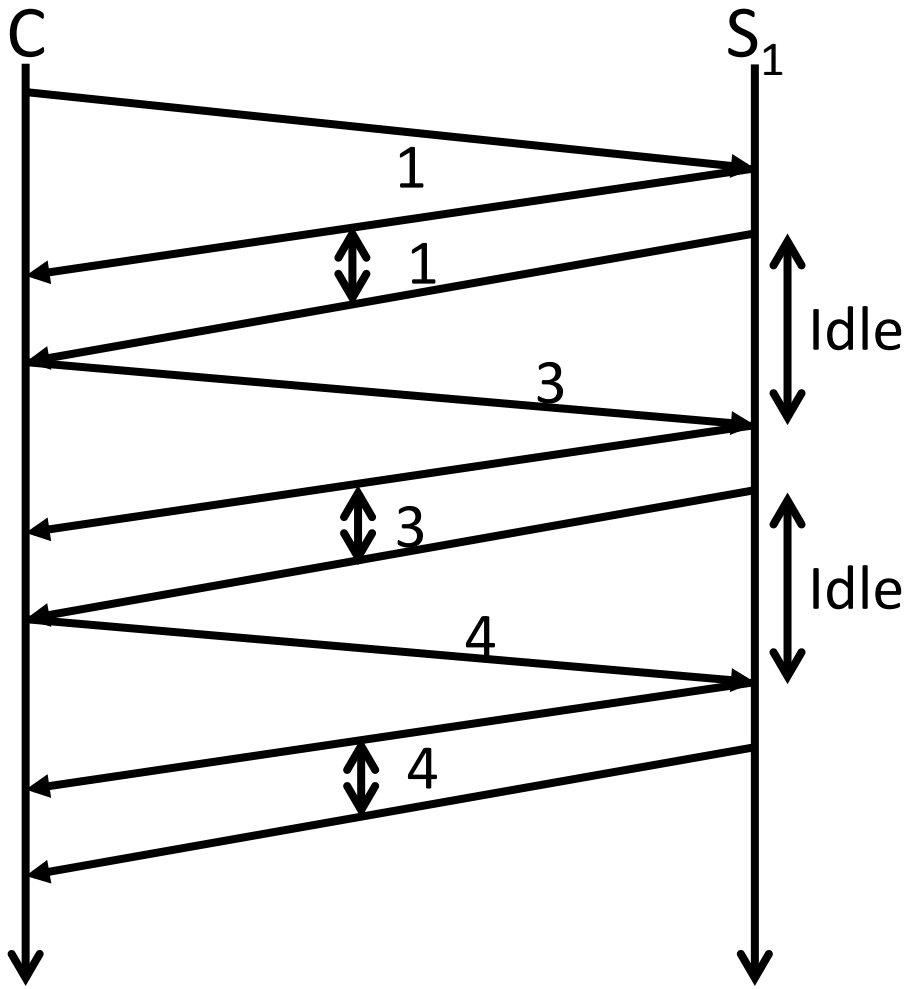
Dynamic Parallel Download: 4 pieces example



Performance Issues

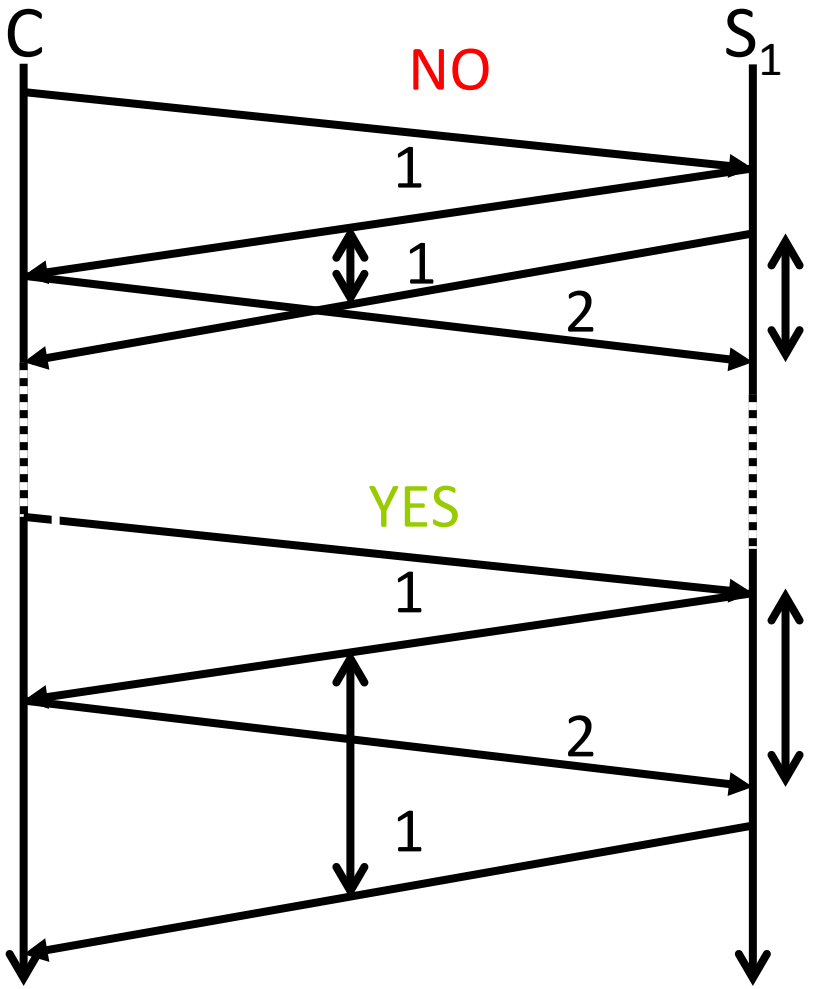
- All servers must be busy sending pieces
- Two performance issues
  - Interblock idle time
    - Pipelining
  - Termination idle time
    - End game mode (Terminology introduced in BitTorrent)

Interblock Idle Time



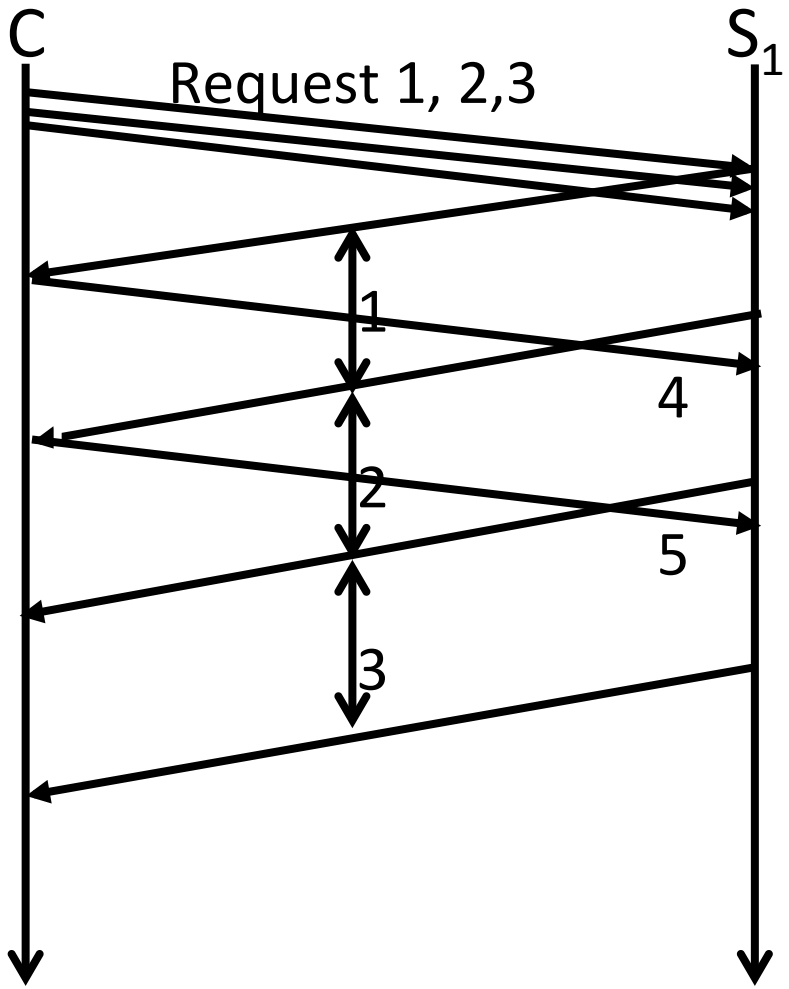
- Time to receive a new request after sending the last byte of a piece
- Idle time = 1 RTT
- Problem
  - Server underutilized
- Solution
  - Pipelining

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



- 2nd 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

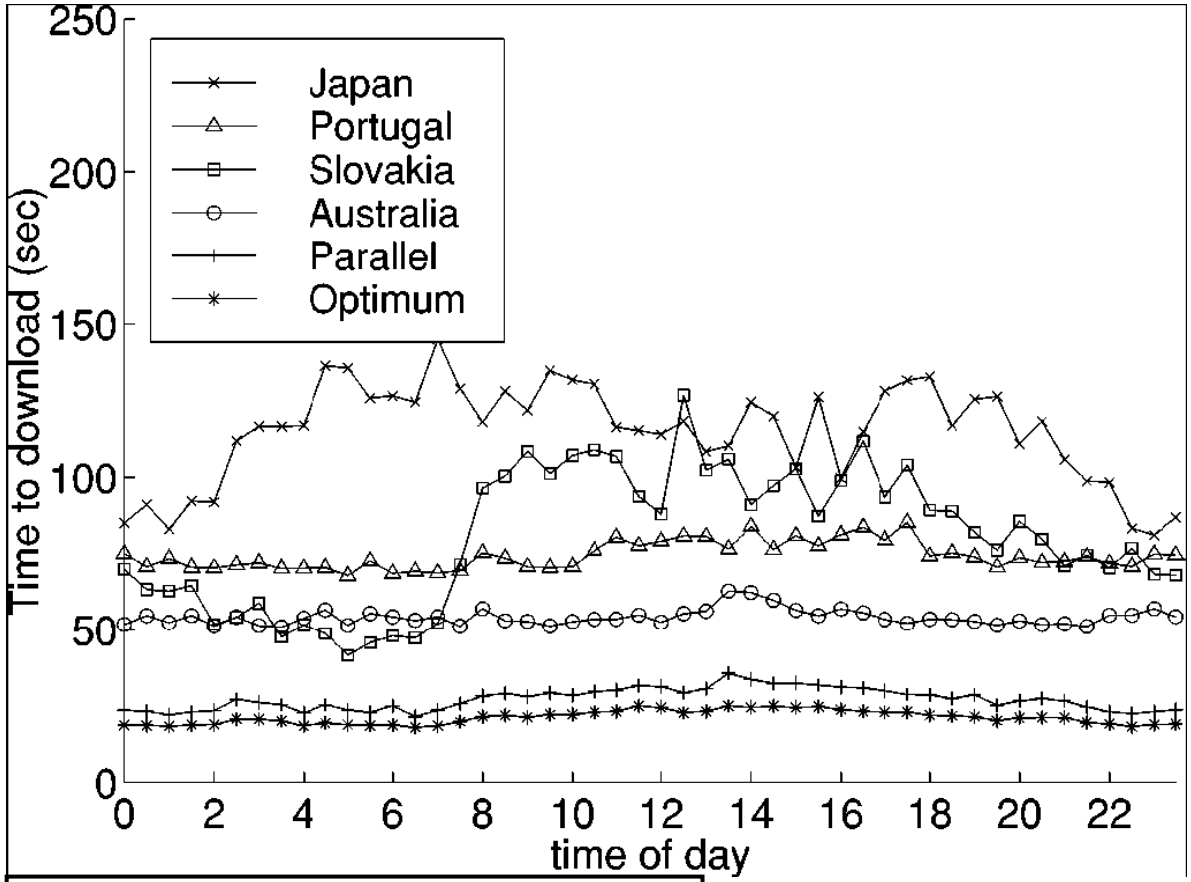
Experimental Evaluation

- Java client that implements dynamic parallel download
  - Does no 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



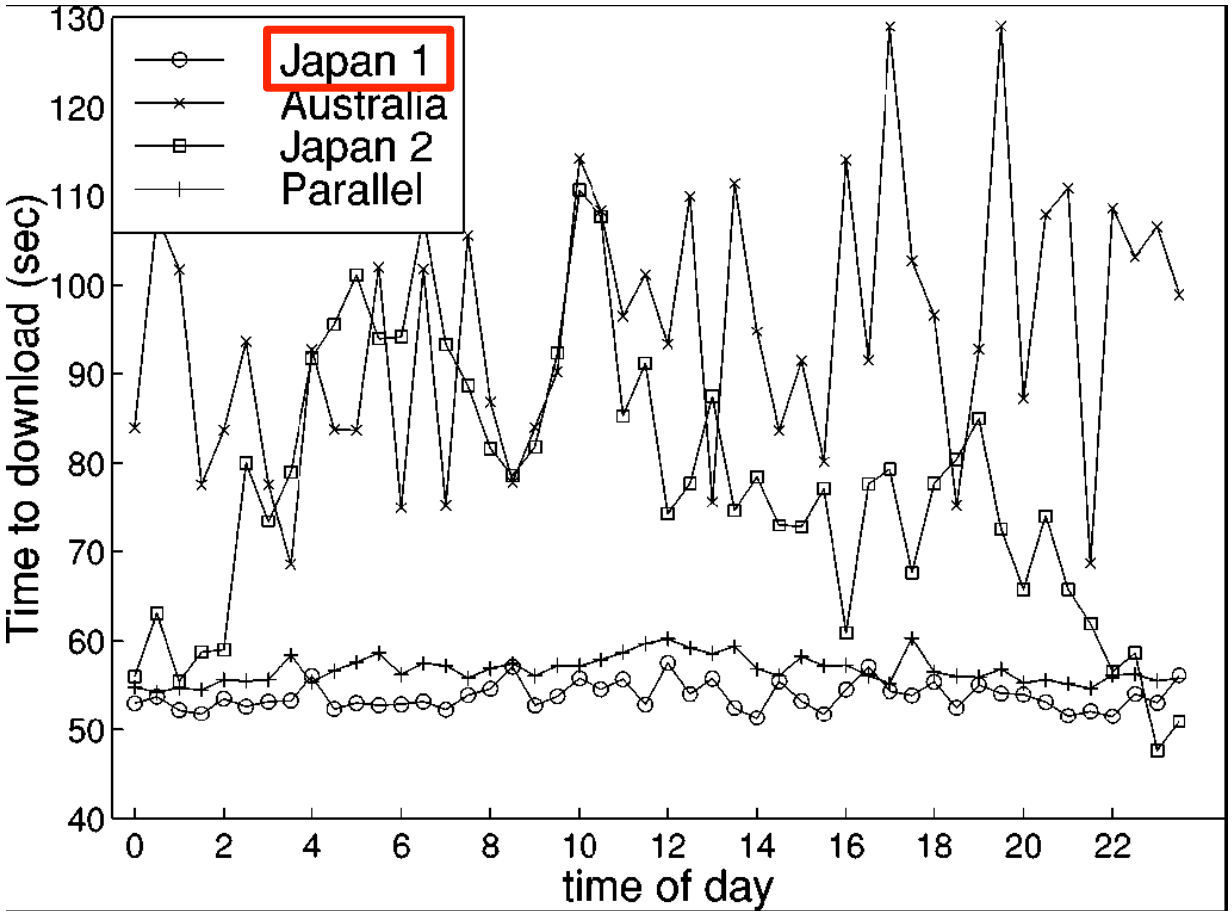
Credit: Rodriguez et al. [8]

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

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



Credit: Rodriguez et al. [8]

- 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

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

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)

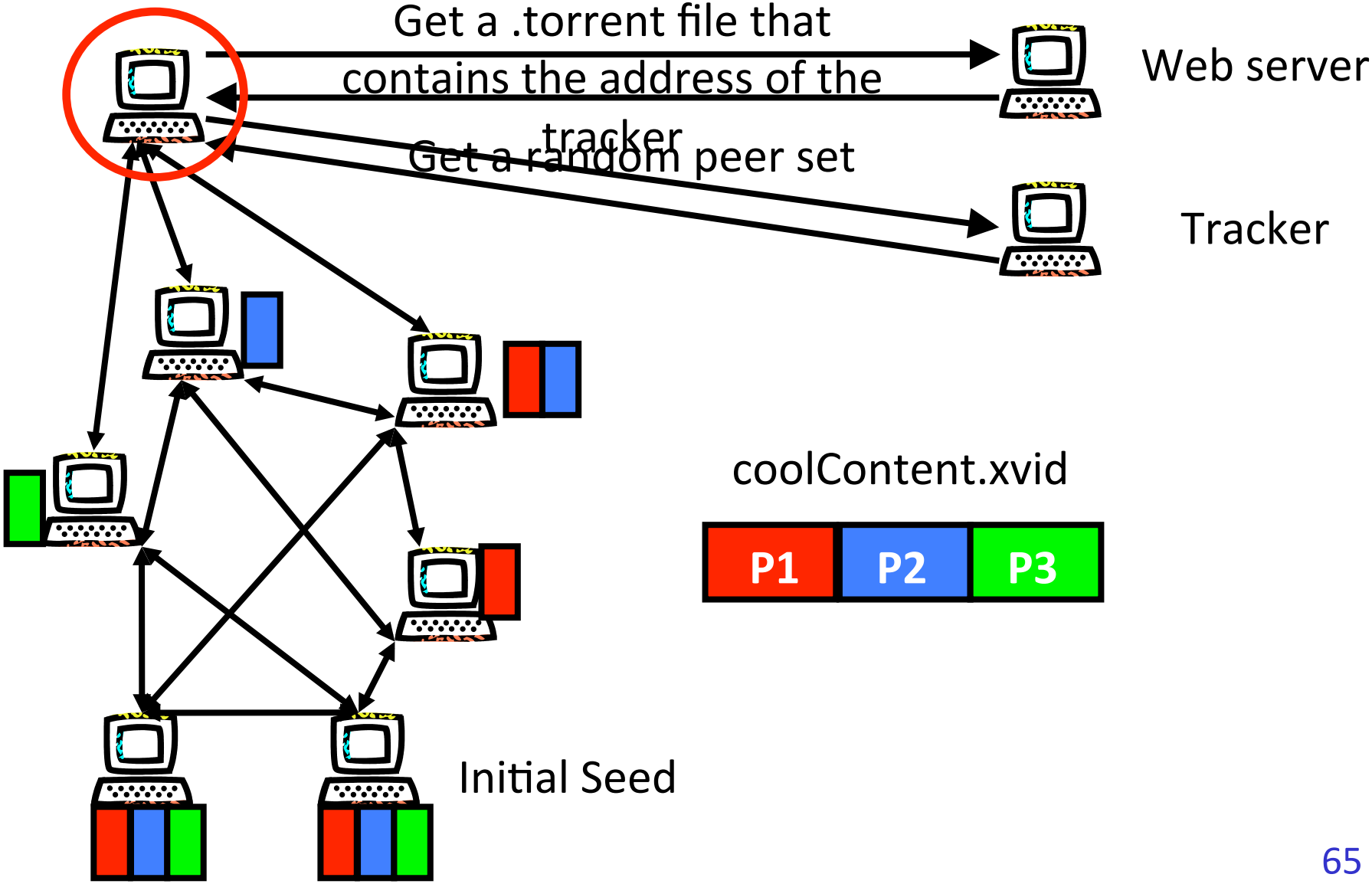
Which Peer and Piece Selection?

- 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 (described in details later in the course)
  - 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

BitTorrent Overview



Transport Protocol

- BitTorrent can use TCP or uTP
- TCP/IP header overhead
  - 40 bytes
- uTP
  - New transport protocol designed by BitTorrent Inc. for uTorrent

uTP

- Why a new transport protocol
  - DSL and cable modems can buffer several seconds worth of packets
    - All traffic crossing this buffer will experience seconds of delays
      - Very bad for interactive applications like VoIP, Web browsing, games, etc.
    - BitTorrent is very aggressive
      - Opens tens of connections

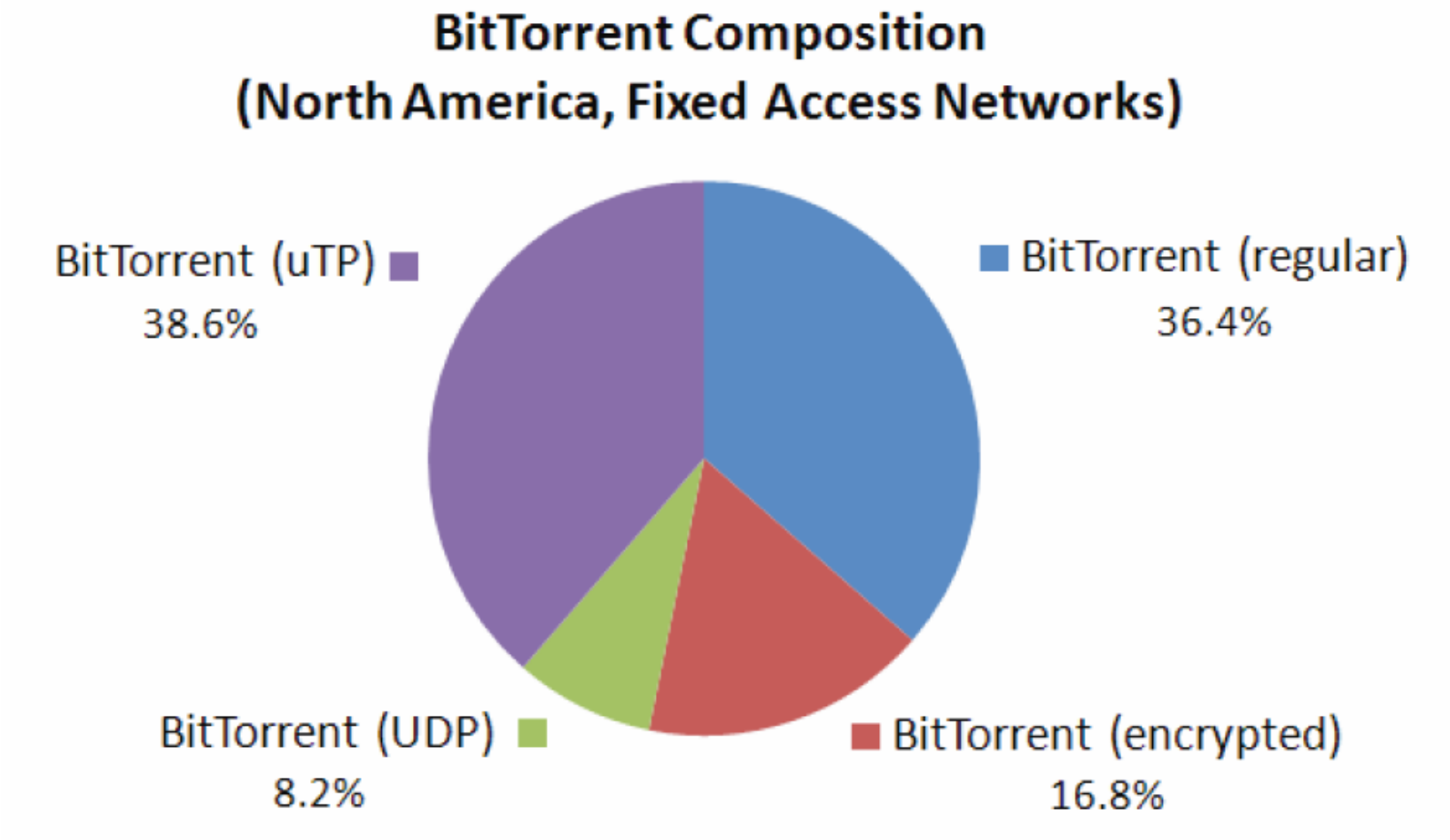
uTP

- Current solution is to cap the upload rate to 80% of the capacity
  - Suboptimal
- uTP solution
  - Use end-to-end delay variations to adapt upload speed
    - When delay increases slowdown
    - When delay decreases speedup
  - Less aggressive than TCP

uTP

- uTP is on top of UDP
  - IP/UDP/uTP overhead
    - 20 bytes for IP
    - 8 bytes UDP
    - 20 bytes uTP
- uTP specification
  - BEP 29

TCP and uTP usage [53]



Credit: sandvine 2011 [53]