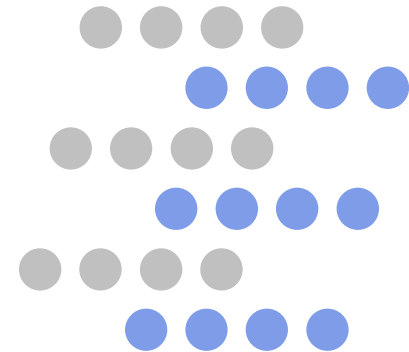


Peer-to-Peer Networks

Analysis of Bit Torrent

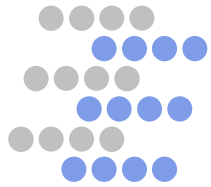


Ernst Biersack

erbi@eurecom.fr

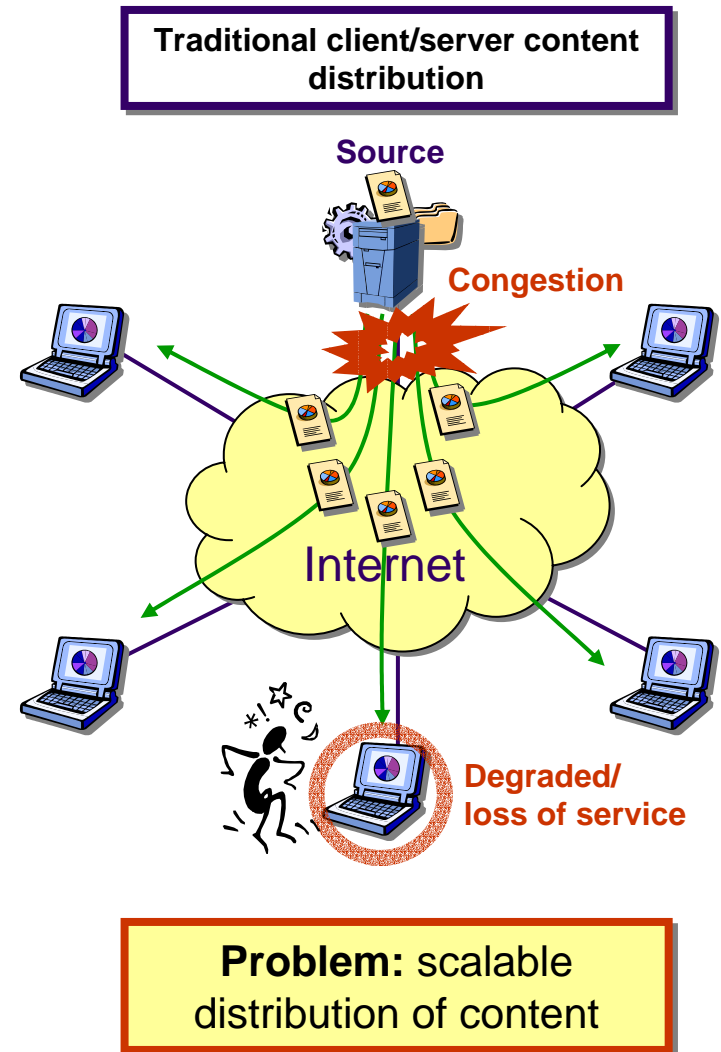
<http://www.eurecom.fr/~erbi/>

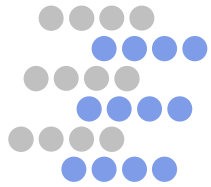
Joint work with M. Izal, G. Urvoy-Keller, and P. Felber



Context and Problem

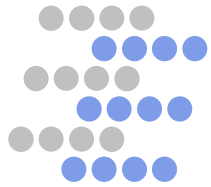
- An growing number of well-connected users access **increasing amounts of content**
- But interest in content is often “Zipf” distributed (small fraction of very popular content)
- Servers and links are **overloaded**
 - Number of clients
 - Size of content
 - “Flash crowd” (e.g., 9/11)
- Tremendous engineering (and cost!) necessary to make server farms **scalable** and **robust**





Real-World Scenarios

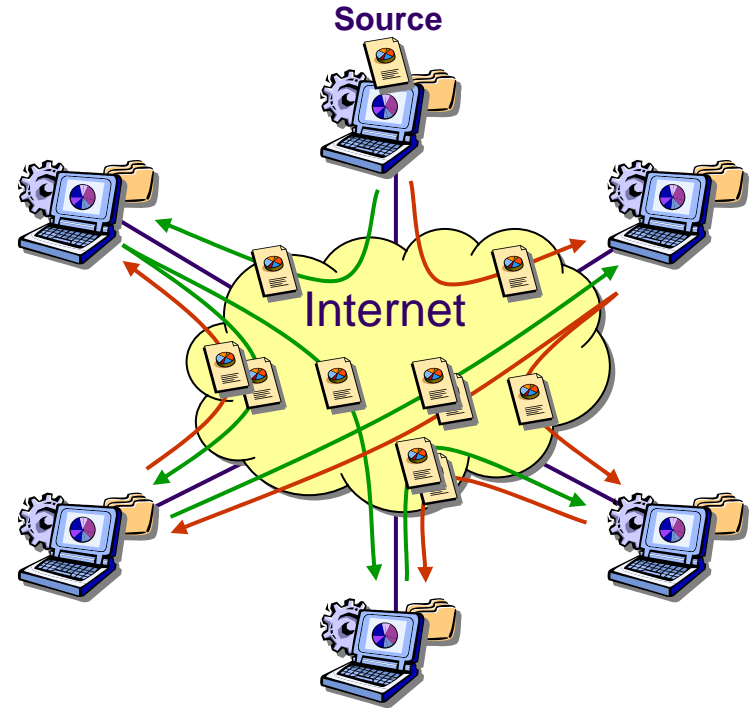
- Quick distribution of critical content
 - E.g., antivirus definitions
- Efficient distribution of large content
 - E.g., nightly update of a bank's branches, promotional movie from manufacturer to all car dealers
- Distribution of streaming content
 - E.g., live event, Internet TV
- Classical approaches have high cost
 - Source over-provisioning (for peak demand)
 - Content Delivery Networks (CDNs)
- **Novel approach:** cooperative networks

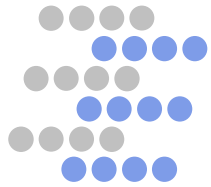


Cooperative Networking

- In a **cooperative network** (“peer-to-peer”), all nodes are both client and server
 - Many nodes, but unreliable and heterogeneous
 - Takes advantage of distributed, shared resources (bandwidth, CPU, storage) on peer nodes
 - Fault-tolerant, self-organizing
 - Dynamic environment: frequent join and leave is the norm

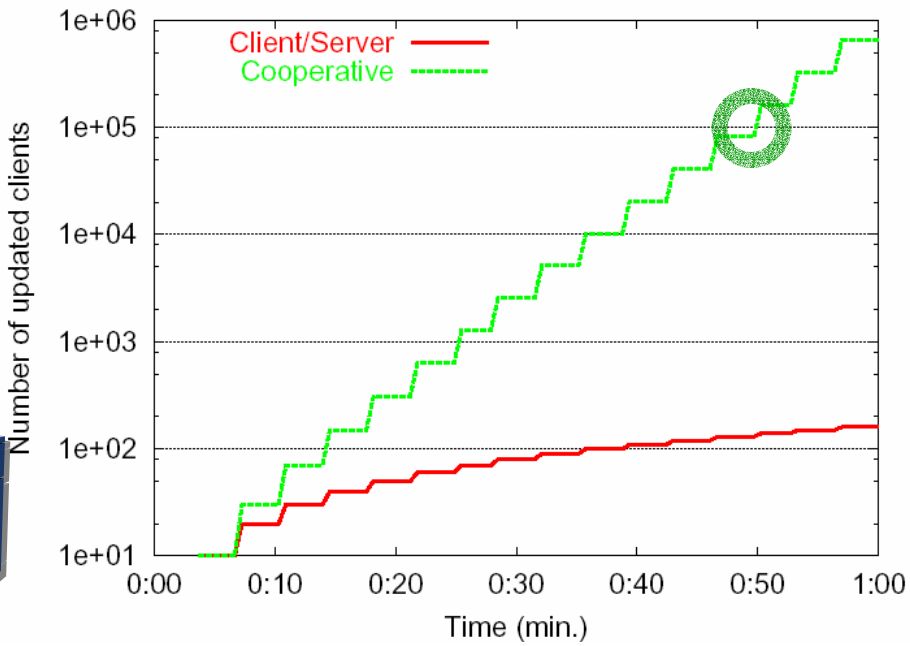
Cooperative content distribution



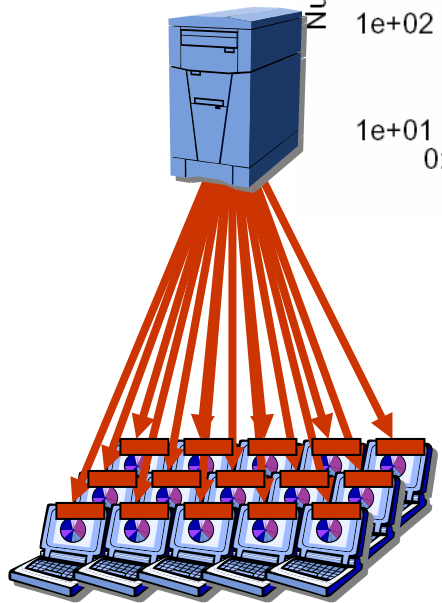


Cooperative Distribution: Intuition

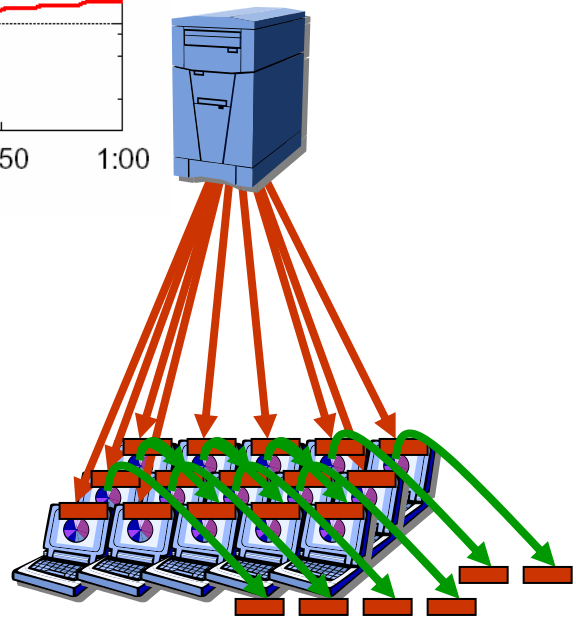
1. 9h:52m
2. 14h:48m



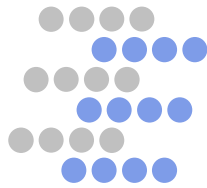
12s
19m:54s



Client/Server

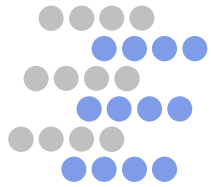


Cooperative



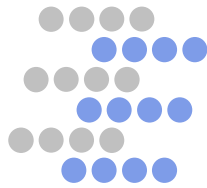
Cooperative Distribution

- **Principle:** Capitalize bandwidth of edge computers
 - **Self-scaling network:** more clients \Rightarrow more aggregate bandwidth \Rightarrow more scalability
 - Cost-effective, robust against failures and flash crowds
- How well does it work in practice?
 - Study of BitTorrent over 5 months, 1.77 GB file, 180,000+ clients
- What are the best cooperative distribution strategies
 - Cooperative network simulations (depending on peer arrivals, bandwidth capacities, peer lifetimes, etc.)



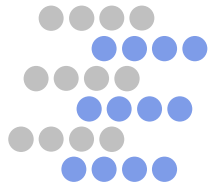
BitTorrent

- Designed for the transfer of large files to many clients
 - Based on **swarming**: a server sends different parts of a file to different clients, and the clients exchange chunks with one another
- One session = distribution of a single (large) file
- Elements
 - An ordinary web server
 - A static “meta-info” file
 - A tracker
 - An initial client with the complete file
 - On end user side: web browser + BitTorrent client

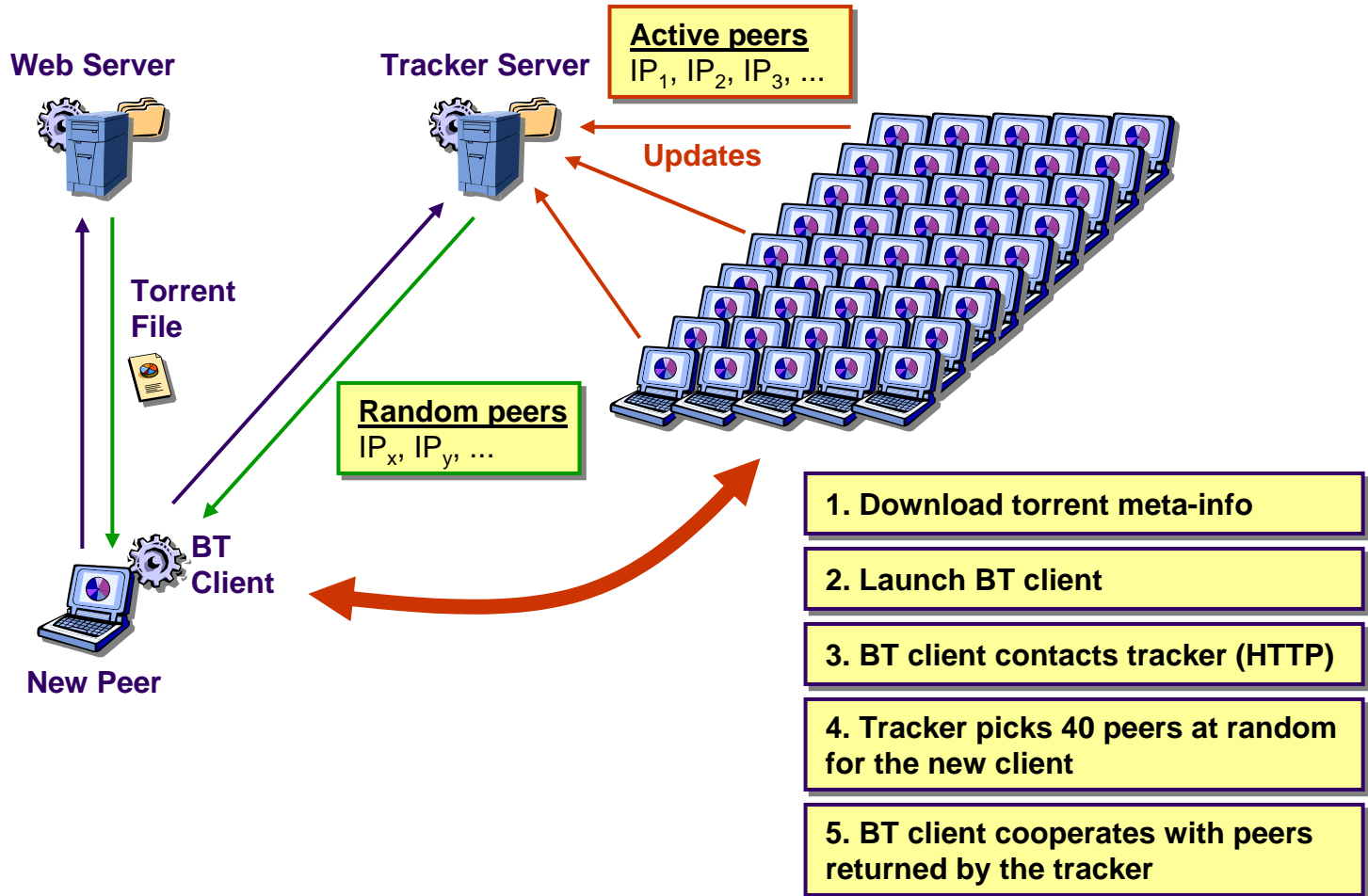


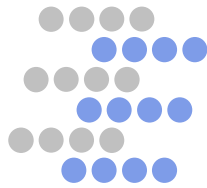
Session Initiation

- Start running a web server that hosts a torrent file
 - The torrent file contains the IP address of the tracker
- The tracker (often not on web server) tracks all peers
 - Initially, it must know at least one peer with the complete file
 - Peer with full file: **seed**
 - Peer still downloading file: **leecher**
- On client side
 - BT client reads tracker IP address and contacts the tracker (through HTTP or HTTPS)
 - The tracker provides to the BT client a set of active peers (leechers and seeds, typically 40) to cooperate with
 - Clients regularly report state (% of download) to the tracker



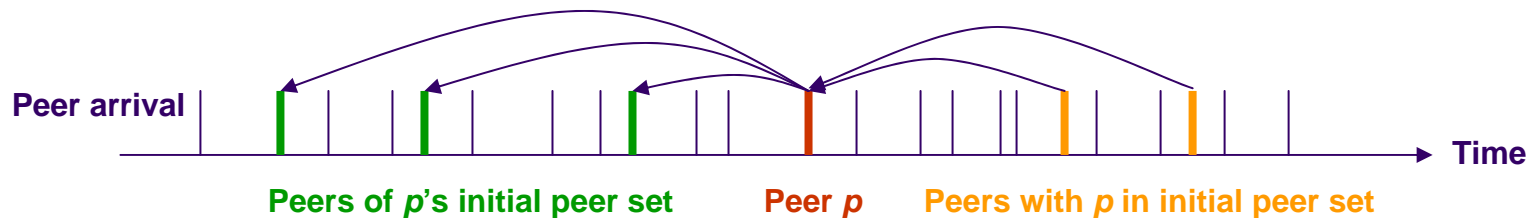
Joining a BT Session

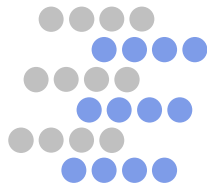




Peer Sets

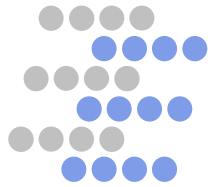
- Tracker picks peers at random in its list
- Once a peer is incorporated in the BT session, it can also be picked to be in the peer set of another peer
- This technique allows a wide temporal diversity
 - A peer knows both **older peers and newcomers!**
 - Ensures transfer of chunks between “generations”
- Note: a peer communicates with its initial peer set and the other peers that contacted it but NOT with other peer sets





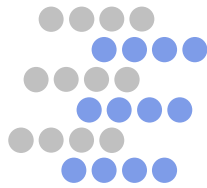
File Transfer Algorithm

- Initial file broken into chunks (typically 256 kB)
 - Torrent file contains SHA1 hash for each chunk: allows to check integrity of each chunk
- Reports sent regularly (at start-up, shutdown, and every 30 minutes) to tracker
 - Unique peer ID, IP, port, quantity of data uploaded and downloaded, status (started, completed, stopped), etc.
- Peer connect with each other over TCP, full duplex (data transit in both directions)
 - Upon connection, peers exchange their list of chunks
 - Each time a peer has downloaded a chunk and checked its integrity, it advertises it to its peer set



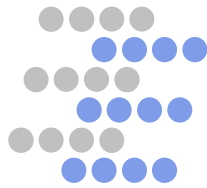
Connection States

- On each side, a connection maintains 2 variables
- **“Interested”**: you have a chunk that I want
 - Allows a peer to know its possible clients for upload
- **“Chocked”**: I don't want to send you data at the time
 - Possible reasons: I have found faster peers, you did not/can't reciprocate enough, ...



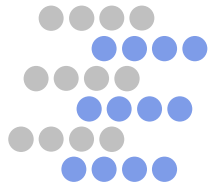
Chunk Selection Algorithm

- Which missing chunk should we request from other peers?
- Simple strategy: **random** selection
 - Choose at random among chunks available in peer set
 - Randomness ensures diversity
- Biased strategy: peers apply the **rarest-first** policy
 - Choose the least represented missing chunk in the peer set
 - Rare chunks can more easily be traded with others
 - Maximize the minimum number of copies of any given chunk in each peer set
- BT uses rarest-first policy except for newcomers that use random to quickly obtain a first block



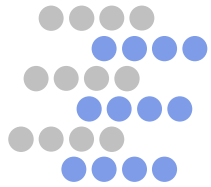
Peer Selection Algorithm

- Serving too many peers simultaneously is not efficient: BT serves 5 hosts in parallel
- Which hosts to serve?
 - The ones that also serve us: **tit for tat** (leechers)
 - The ones that offer the **best download rates** (seeds)
- Can there be any better hosts?
 - Optimistically unchoke a random peer every 30 s to give a chance to another host to provide better service
 - Newcomers have less data to offer \Rightarrow give them “priority” in the optimistic unchoke
 - BT reconsiders choking/unchoking every 10 s (long enough for TCP to reach steady state)



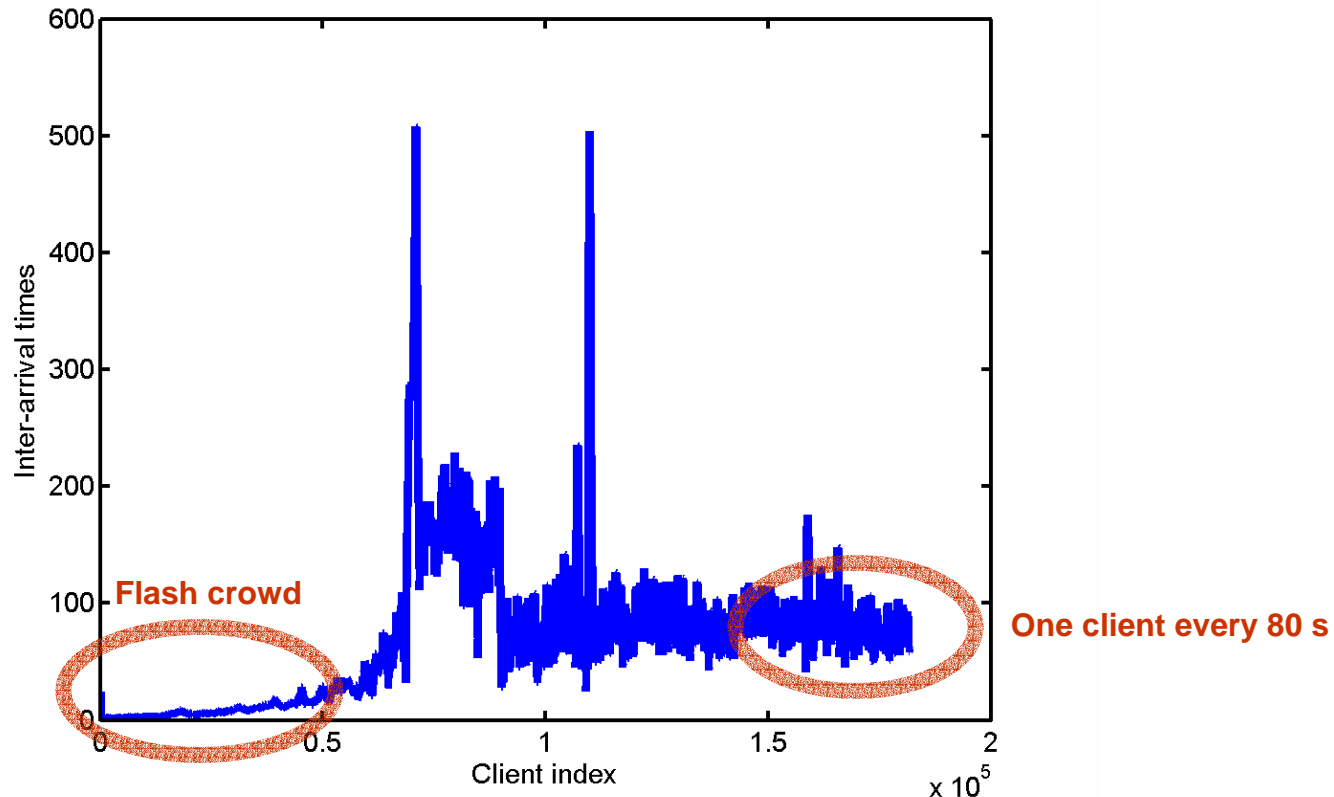
BitTorrent Study

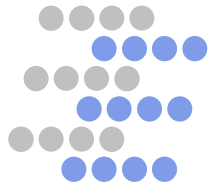
- Five months (April to August 2003) **tracker log** of a very popular BT session
 - Linux RedHat 9, 1.77 GB file
 - Log contains all the reports of all the clients (ID, IP, amount of bytes uploaded and downloaded)
- In addition, we ran our own **instrumented client** on 3 different days to observe a given peer set
 - Log contains blocks uploaded to and downloaded by each host (each time a host has a new block, it advertises its peer set)
 - Exhibits the behavior of BT during the download phase and once the client becomes seed



Tracker Log

- 180,000 clients during the 5 five months period
- Initial flash crowd: 51,000 clients during the first 5 days

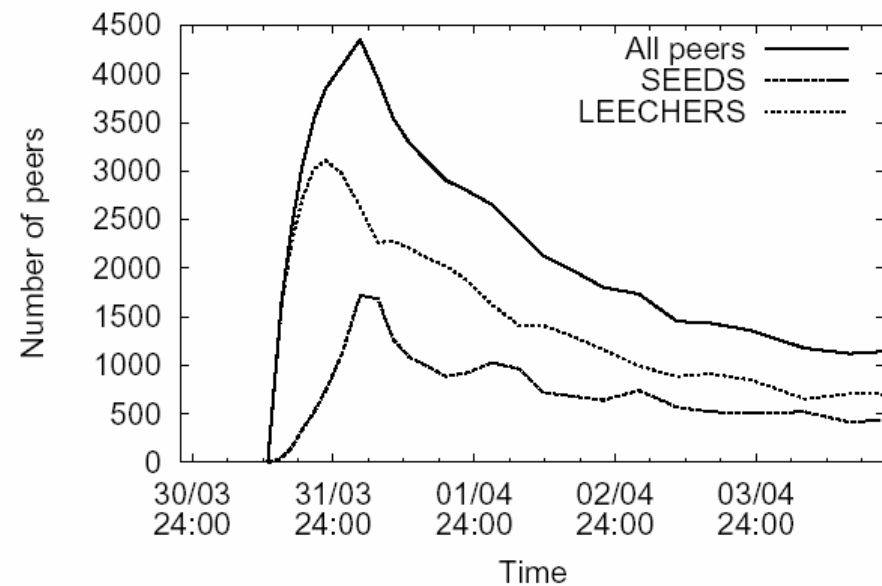
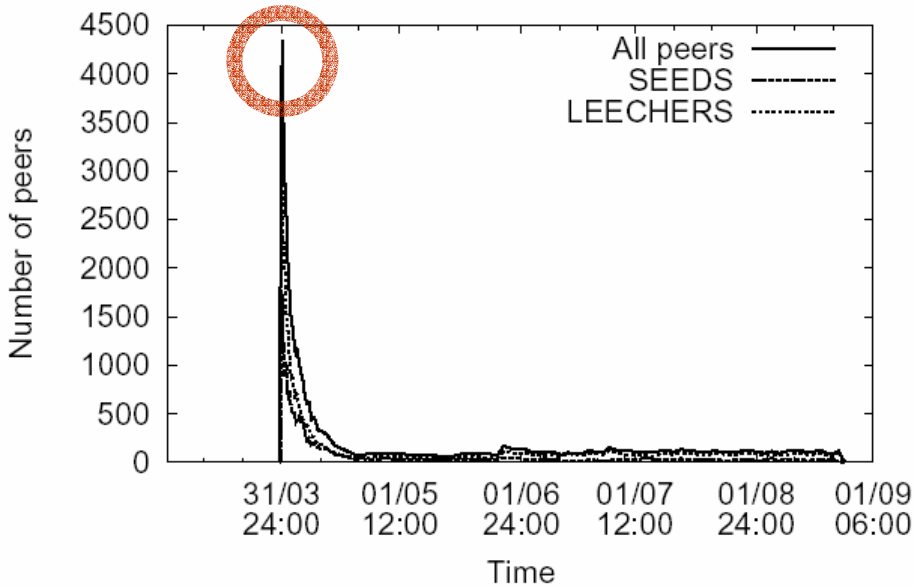


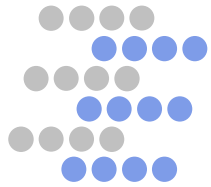


Tracker Log: Number of Clients

- Reaches 4000+ active clients in the first day
- Remains in the interval [100,200] later

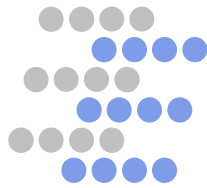
Flash crowd





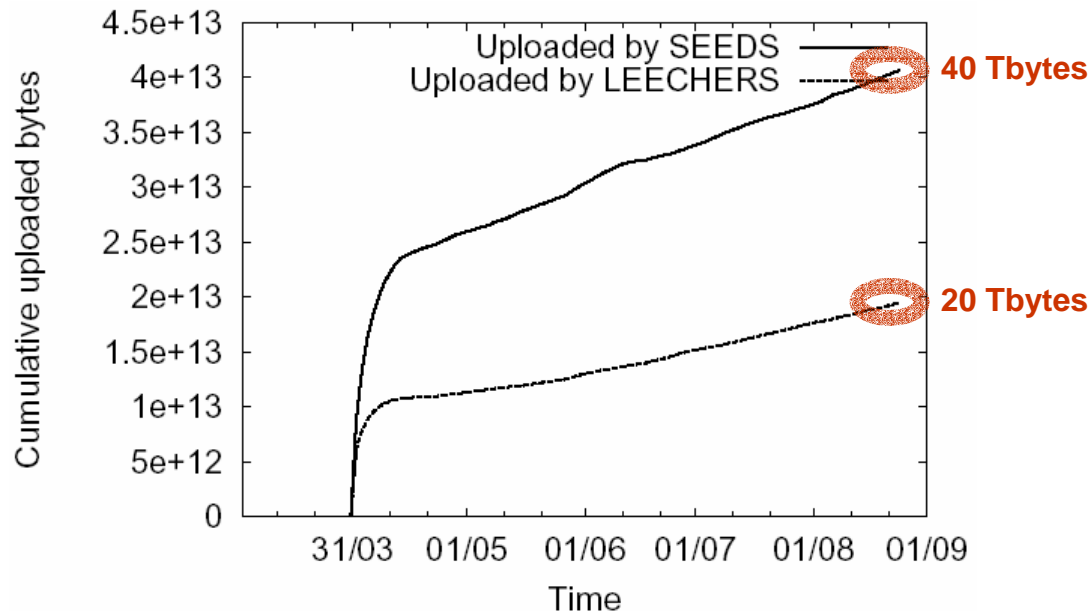
Tracker Log: Clients' behavior

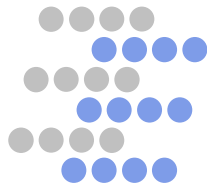
- Clients are very altruistic
 1. When they are leechers
 - They have no choice due to tit-for-tat
 2. Once download is completed since they stay on average **3 hours** after download
 - The transfer is long, may complete overnight
 - The content is legal (RIAA will not sue!)
 - The user is very kind



Tracker Log: Seeds

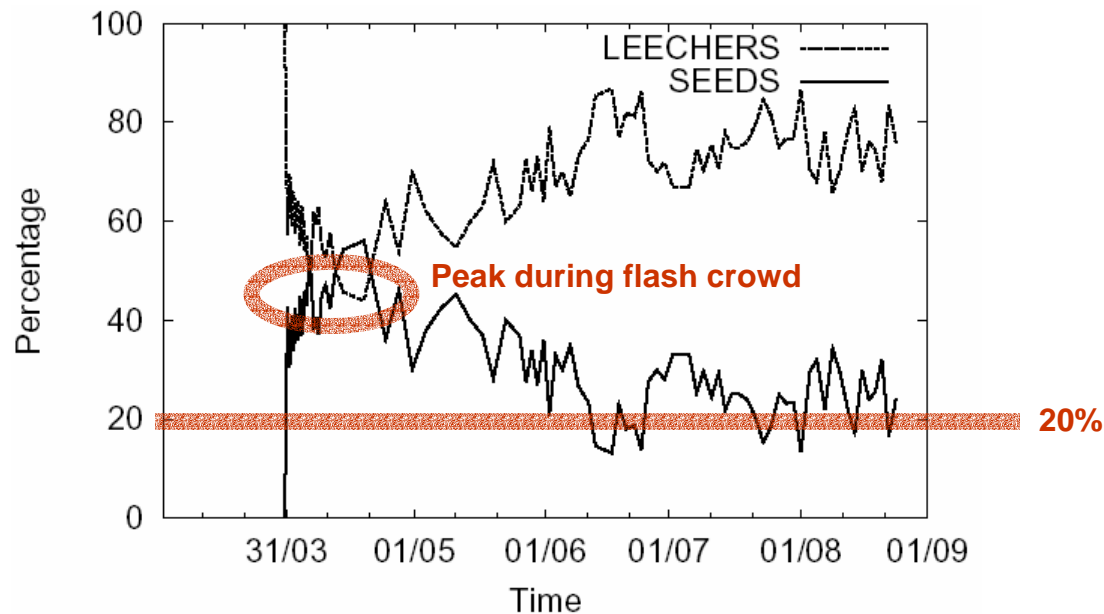
- Presence of seeds is a key feature of BT
 - Over the 5 months they contributed twice as much volume as leechers

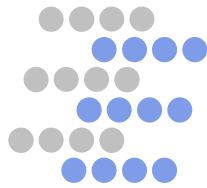




Tracker Log: Seeds vs. Leechers

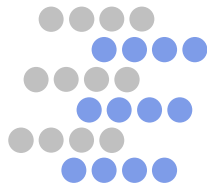
- The percentage of seeds is consistently high (20+%)
- Thus, two factors allow BT to sustain the flash crowd:
 - Its ability to quickly create seeds (i.e., complete downloads)
 - The fact that users are altruistic and seeds remain online





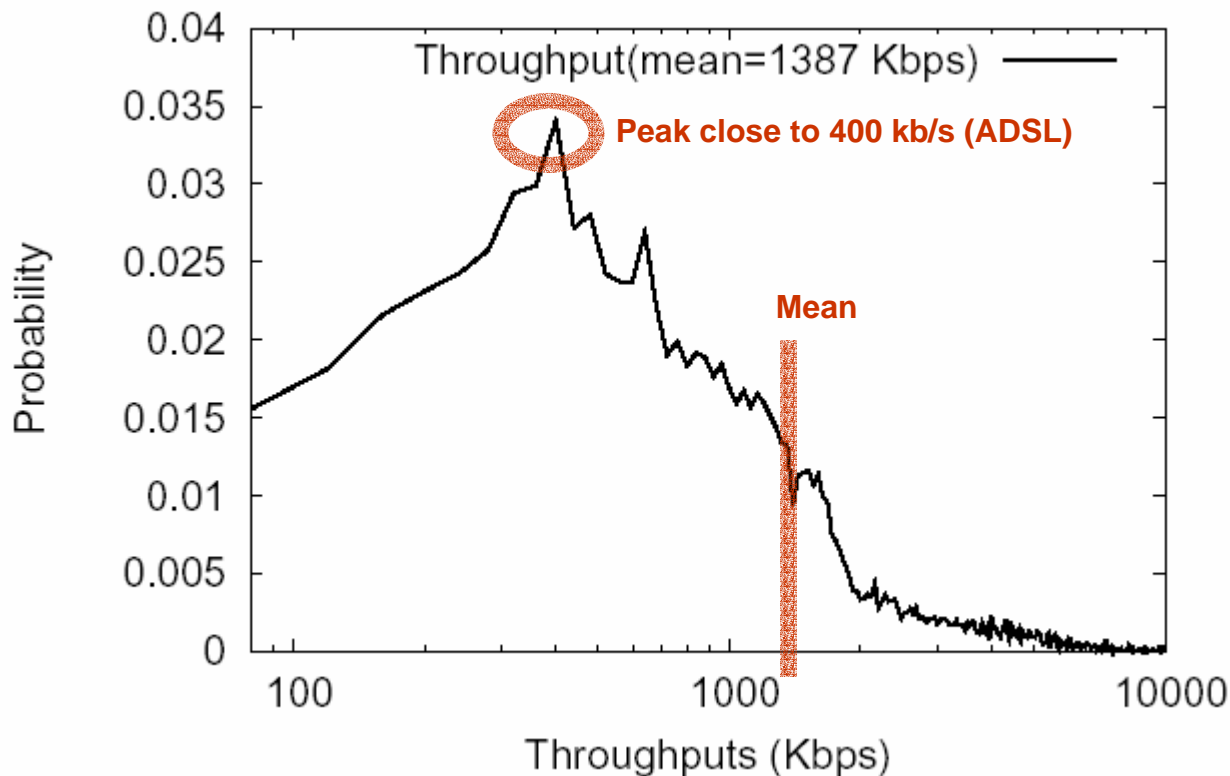
Tracker Log: BT vs. Mirroring

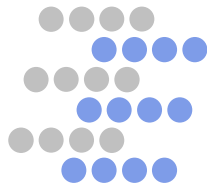
- Throughput per leecher is always **above 500 kb/s**
 - At least ADSL client
- Aggregate throughput of system (sum over all leechers at each instant) was **higher than 800 Mb/s**
 - More than 80 mirrors, each sustaining a 10 Mb/s service
- Considering only the 20,000 hosts that completed download in a single session (BT allows resume)
 - Throughput is better than average: **1.3 Mb/s**
 - Average download time is **30,000 s** (8.3 h)



Tracker Log: Complete Sessions

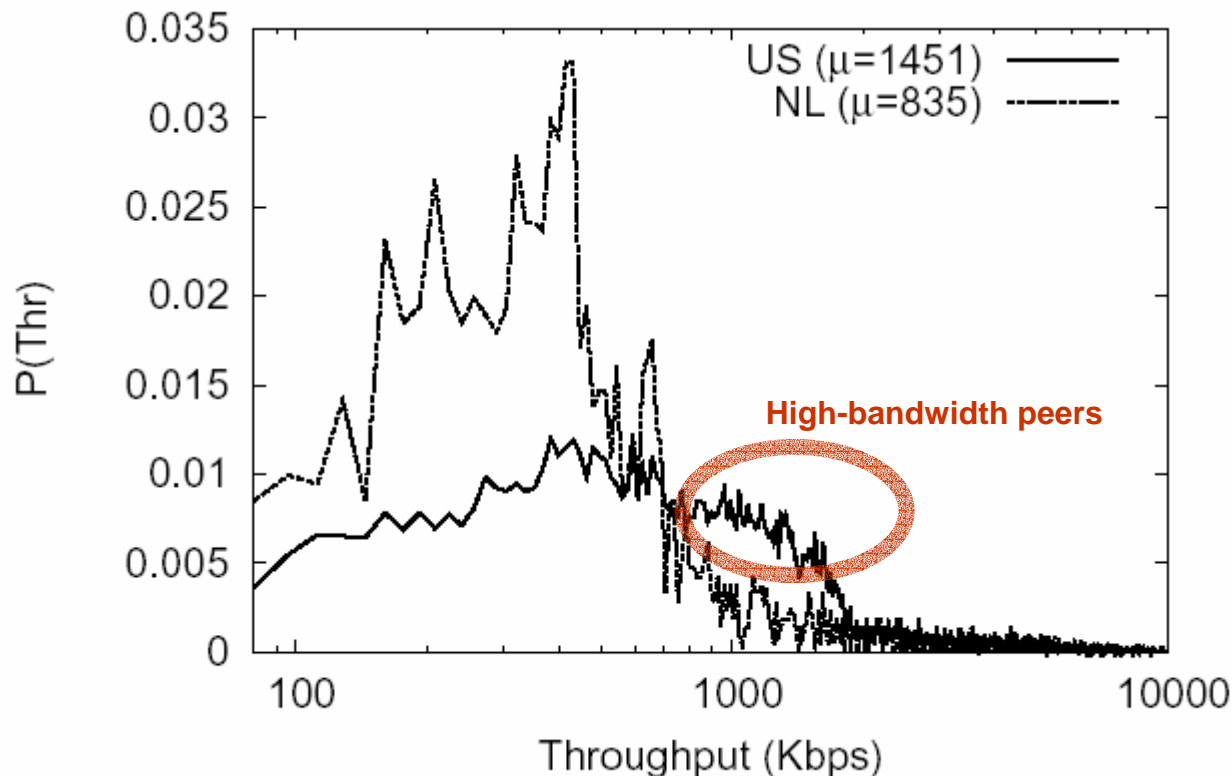
- Peak value around ADSL speed
- Some hosts have very high bandwidth

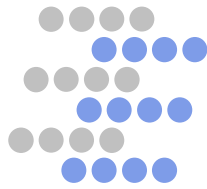




Tracker Log: US vs. Europe

- In the first 4 weeks:
 - US clients have better access links than European clients





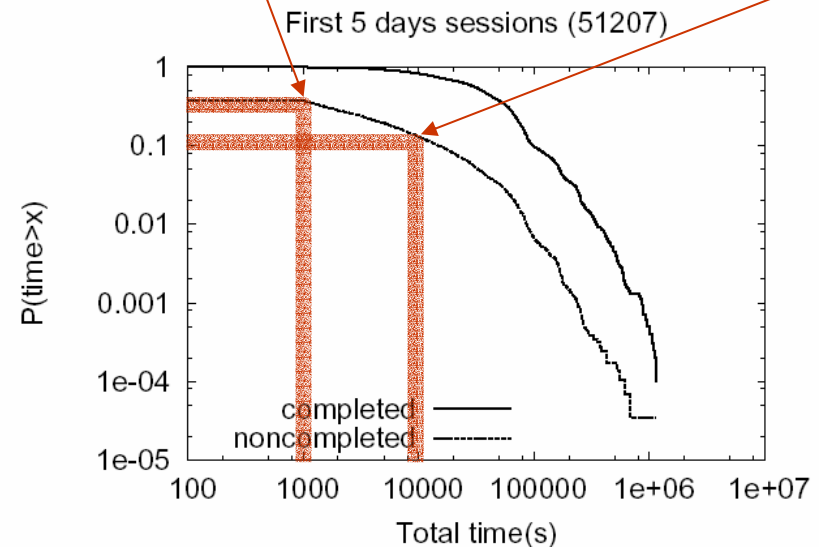
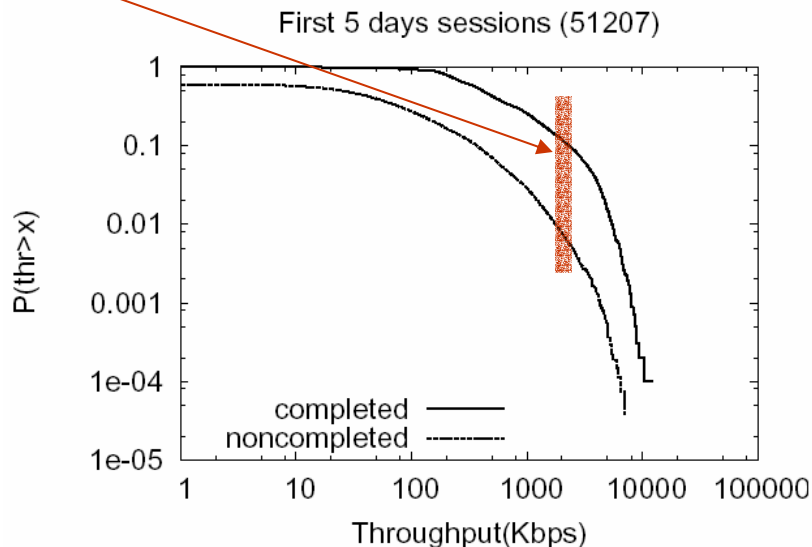
Tracker Log: Incomplete Sessions

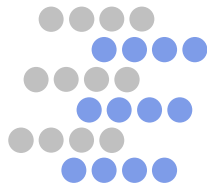
- Causes of abortion (no interest, crash)?
 - Assumption: abortions due to experiencing bad service
 - Valid if users receive almost nothing while online

Throughput of incomplete sessions smaller than that of complete sessions

60% of the incomplete sessions last less than 1,000 s (<20 m)

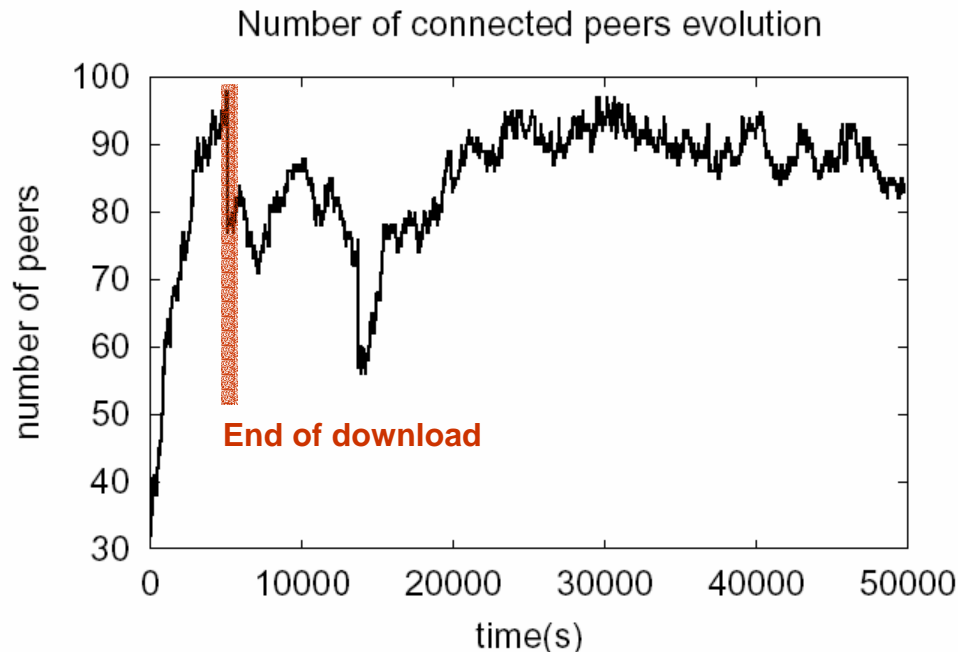
90% of the incomplete sessions last less than 10,000 s (<3 h)



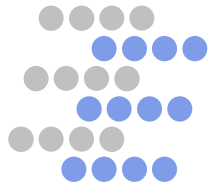


Client Log

- Modified client behind a 10 Mb/s campus access link
 - 3 transfers, 3 days of 5th month (far from flash-crowd)
 - Average transfer time: **4,500 s** (1.25 h, fast client!)
 - We remained as seed for another 13 hours



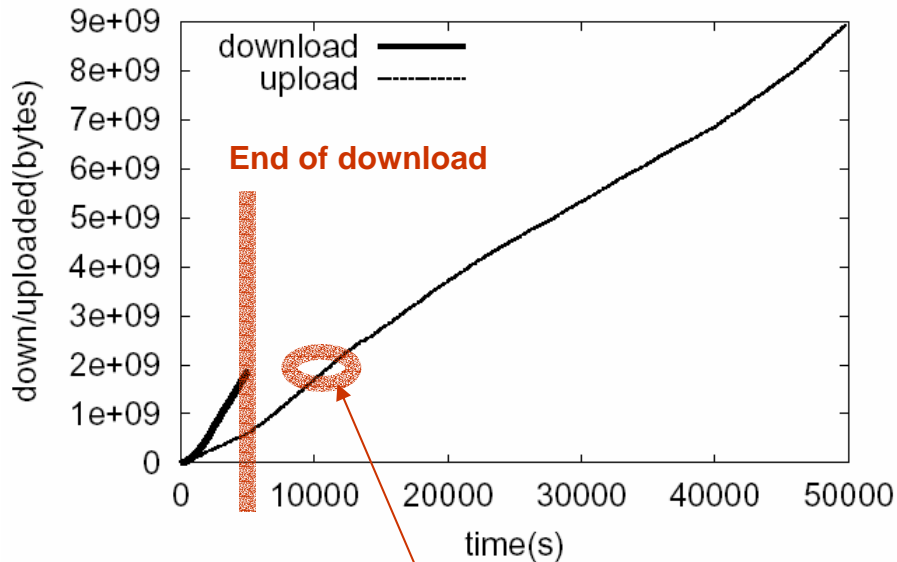
Number of clients drops after end of download phase.
Explanation: seeds disconnect



Client Log: Upload and Download

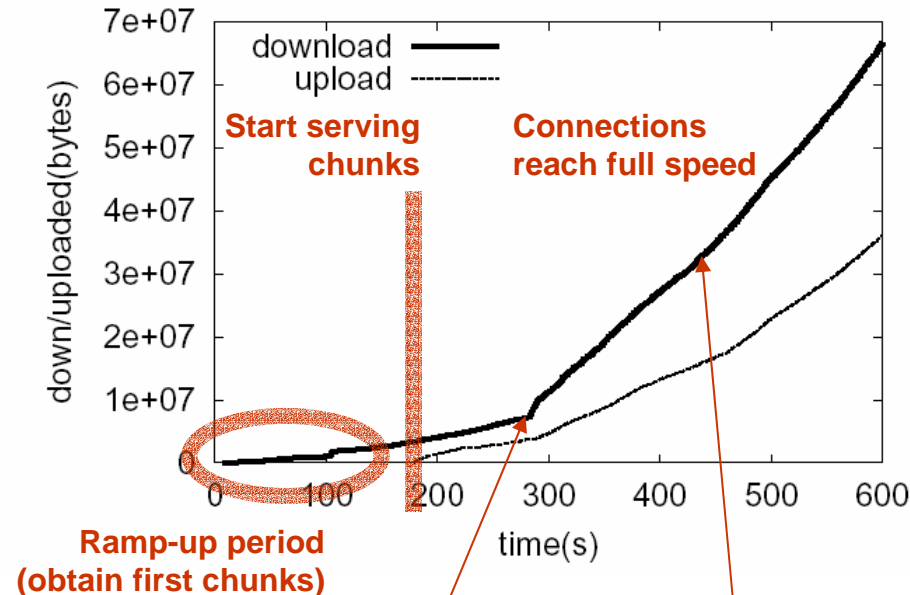
Client never gets stalled: we always find peers to serve and download chunks from ⇒ **good efficiency**

Cumulative download and upload evolution

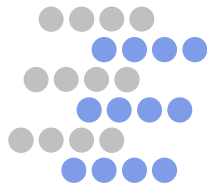


We uploaded as much as we downloaded after 10,000 s = **twice the download time**

Cumulative download and upload evolution

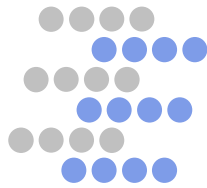


Cooperation is enforced: the download rate increases **because** the upload rate increases



Client Log: Tit-for-Tat

- Who gave us the file, seeds or leechers?
 - 40% from seeds and 60% from leechers
 - 85% of the file was provided by only 25% peers
 - Most of the file provided by peers that connected to us (not from original peer set)
- How good is the tit-for-tat policy?
 - Two conflicting goals
 - Must enforce cooperation among peers
 - Must allow transfer even if bandwidth not perfectly balanced



Client Log: Tit-for-Tat

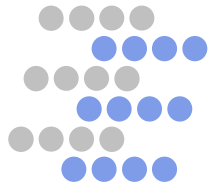
- We found that BT pays more attention to the amount of data transferred than to the balance of bandwidths — **a very good property**

High correlation of traffic volumes
Low correlation of throughputs

Correlation coefficient	Trace		
	14th	15th	18th
$Corr[V_d, T_d]$	0.917915	0.789044	0.974185
$Corr[V_u, T_u]$	0.462702	0.34648	0.50432
$Corr[V_d, V_u V_d \cdot V_u \neq 0]$	0.434974	0.730305	0.432803
$Corr[T_d, T_u T_d \cdot T_u \neq 0]$	-0.165767	-0.032086	-0.256644

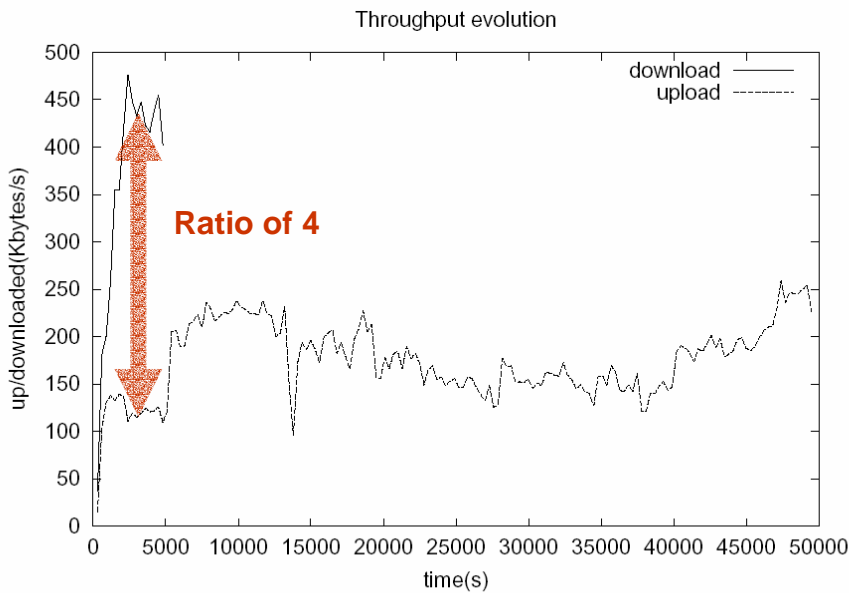
Legend:
V: volume
T: throughput
d: download
u: upload

Table 3: Correlation coefficients between per connection variables

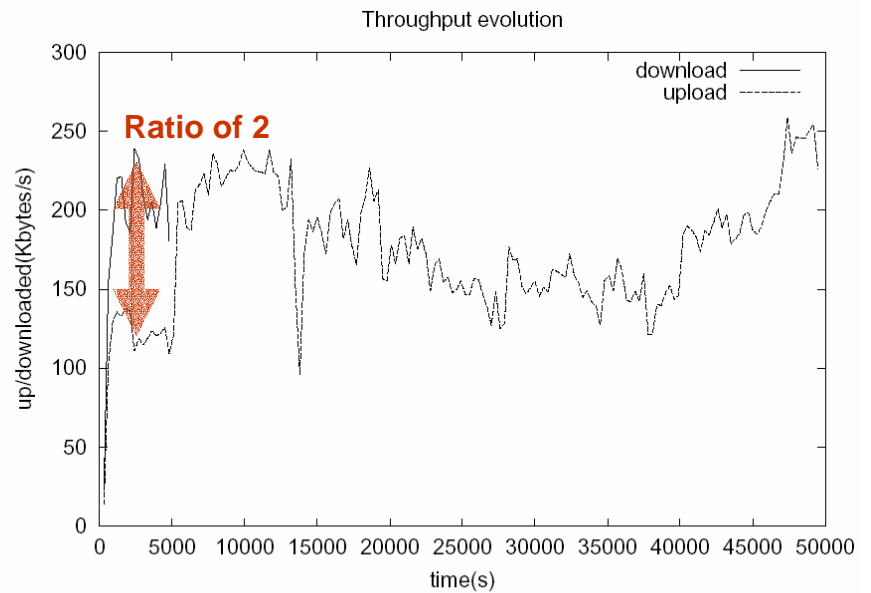


Client Log: Tit-for-Tat

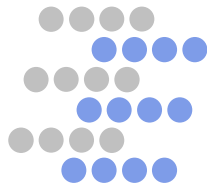
- We received more than we gave, even if we do not account for seeds traffic
 - Probably due to our good download capacity and to tit-for-tat enforcement



(a) All peers



(b) Non-seeds only



BitTorrent Summary

- BT seems very efficient for highly popular downloads
 - Still, its performance might be affected if clients do not stay long enough as seeds, e.g., in case of illegal content...
 - What happened to 160,000 incomplete downloads?
- BT is clearly able to sustain large flash crowds
- Some opened questions
 - Could we do better by using different peer and chunk selection strategies?
 - How would BT do if all peers arrive at the same time (e.g., antivirus update)?
 - Could we do better if peers have symmetric bandwidth (e.g., private network)?

Questions?

