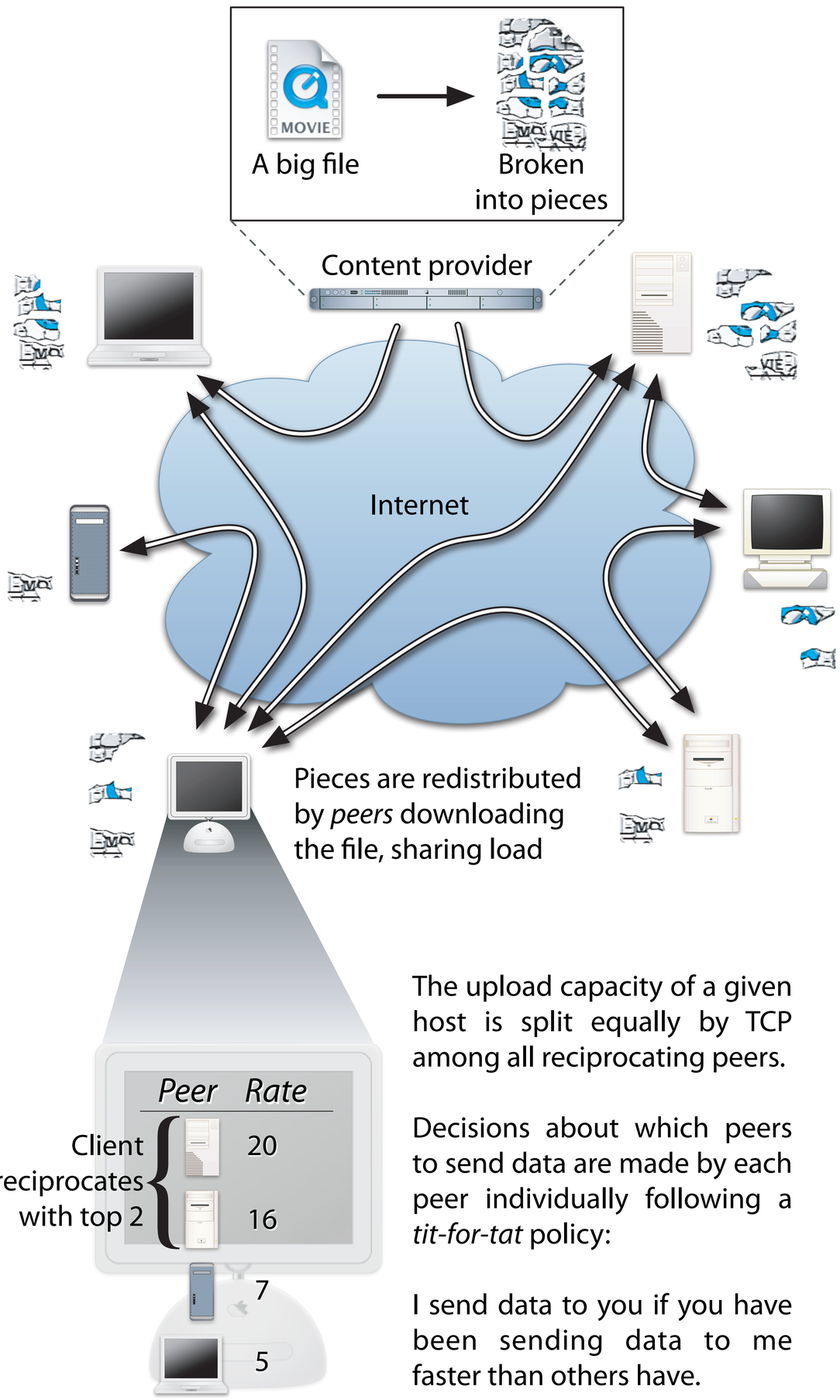
Do incentives build robustness in BitTorrent?

Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, Thomas Anderson

Overview

A fundamental problem with many peer-to-peer systems is the tendency of users to "free ride"—consume resources without contributing to the system. The popular file distribution tool BitTorrent was explicitly designed to address this problem, using a tit-for-tat reciprocity strategy to provide positive incentives for users to contribute resources to the system. We show that although BitTorrent has been fantastically successful, its incentive mechanism can be cheated by selfish clients.

How BitTorrent works today



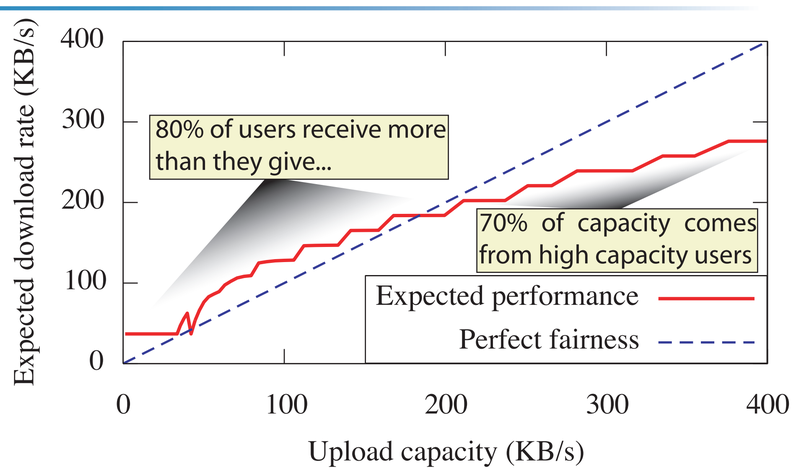
The upload capacity of a given host is split equally by TCP among all reciprocating peers.

Decisions about which peers to send data are made by each peer individually following a *tit-for-tat* policy:

I send data to you if you have been sending data to me faster than others have.

Fairness

Ideally, tit-for-tat provides fairness: each person receives data as quickly as they contribute. In practice, high capacity users contribute much more than they receive.



Cheating with *BitTyrant*

The unfairness of BitTorrent suggests that tit-for-tat does not work as intended and might be exploited by selfish users to improve performance. We have built BitTyrant, a selfish client designed to do exactly this.

Key idea: BitTyrant dynamically chooses *how many* and *which* peers to send data. In contrast, existing BitTorrent clients send data to a fixed number of peers each tit-for-tat round, regardless of upload capacity.

Our dynamic adjustment algorithm maintains estimates of the rate at which peers will provide data, d, and the rate required to earn reciprocation, u. Using these estimates, we select the highest capacity peers and send them data at the minimum rate that will cause them to reciprocate.

Each round, rank order each peer p by the ratio d_p/u_p, and choose those of top rank until the local upload capacity is reached.

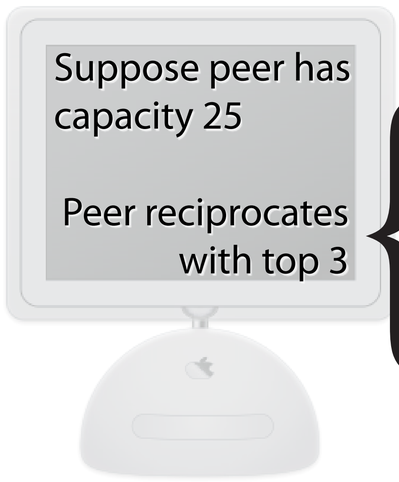
$$\underbrace{\frac{d_0}{u_0}, \frac{d_1}{u_1}, \frac{d_2}{u_2}, \frac{d_3}{u_3}, \frac{d_4}{u_4}}_{\text{choose } k \mid \sum_{i=0}^{k} u_i \leq \text{capacity}} , \ldots$$

At the end of each round for each unchoked peer:

If peer p does not send data: increase cost estimate, u_p.

If peer p has unchoked us for the last minute: reduce cost estimate, u_p.

In our example at left, an existing BitTorrent client might unchoke two peers based on observed received rate only. If the client had 25 KB/s of available capacity, each peer would receive data at 12.5 KB/s. In contrast, BitTyrant can determine which peers are best to exchange with and how many can be supported.



Peer	Received rate	Required send rate	Benefit/cost ratio
	5	2	2.50
	20	16	1.25
	7	7	1.00
	16	16	1.00

Results

We have compared performance of BitTyrant and existing BitTorrent implementations on more than 100 real-world swarms as well as synthetic swarms on the PlanetLab testbed.

- On real swarms, BitTyrant improves download performance by 70% compared to existing BitTorrent clients. Some downloads finish more than 3 times as quickly. Regardless of capacity, using BitTyrant is in the selfish interest of every peer individually.

- However, when *all* peers behave selfishly, average performance degrades for all peers, even those with high capacity.