Bittyrant

Standard bittorrent has an optimistic TFT strategy

- set of peers to which a client sends == active set
- a client sends to
  - unchoked peers: peers from which it received data most rapidly
  - optimistically unchoked: bootstrap new peers, probe for better sources
- if a peer doesn't send data quickly enough to earn reciprocation, it is choked
- a peer splits its upload bandwidth equally across unchoked peers
  - "equal split rate"
    - equals upload capacity / size(active set)
  - active set proportional to sqr_root(upload capacity) by default

Idea is to force peers to earn their way

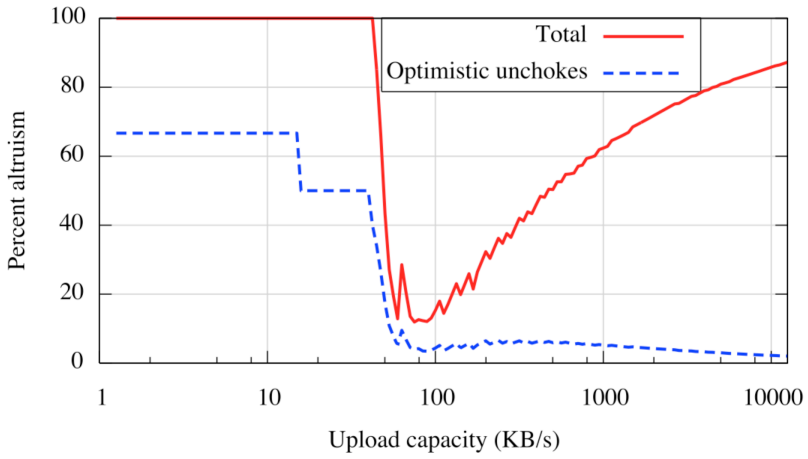
- if I upload to you, I'll be unchoked by you, and I'll get data from you
- "tit for tat"

But, note a few subtleties

- nothing guarantees rate matching
  - standard clients saturate their upload capacity
  - likely means altruism
- nothing smart about upload allocation
  - equal split rate doesn't proportionally reward peers
  - likely inefficient

Bittyrant

Notes a big fraction of upload capacity is spent altruistically – i.e., in a manner that doesn't result in reciprocation. (in other words, that bandwidth is wasted from a greedy point of view.)



Goal of bittyrant: maximize reciprocation, in order to maximize DL rate

- maximize reciprocation bandwidth per connection
  - find peers that use "equal split rate" and have high UL capacity
- maximize number of reciprocating peers
  - expand active set to maximize # of reciprocators, until benefit of additional peer is outweighed by cost of reduced reciprocation probability from other peers
- deviate from equal split
  - lower contribution to a peer as long as it continues to reciprocate
  - saved bandwidth could be allocated to new connections

Go over unchoke algorithm:

For each peer p, maintain estimates of expected download performance d_p and upload required for reciprocation u_p.

Initialize u_p and d_p assuming the bandwidth distribution in Figure 2.

d_p is initially the expected equal split capacity of p.

u_p is initially the rate just above the step in the reciprocation probability.

Each round, rank order peers by the ratio d_p/u_p and unchoke those of top rank until the upload capacity is reached.

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

At the end of each round for each unchoked peer:

If peer p does not unchoke us: $u_p \leftarrow (1+\delta)u_p$

If peer p unchokes us: $d_p \leftarrow$ observed rate.

If peer p has unchoked us for the last r rounds: $u_p \leftarrow (1-\gamma)u_p$

Figure 9: *BitTyrant* unchoke algorithm

A side-effect of this is that the algorithm dynamically discovers the point of diminishing returns for contributing bandwidth

- a selfish peer can withhold bandwidth beyond this

If everybody uses bittyrant, performance gets worse only if peers act selfishly