Users want P2P, we make it work

Stanislav Shalunov <shalunov@bittorrent.com>
Eric Klinker <eklinker@bittorrent.com>

IETF P2P Infrastructure Workshop
Boston, May 28, 2008

Good news

- Content delivery cheaper
- More content
- More bits consumed
- More equipment made
- **More Internet**

Bad news

- End-user RTT in **seconds**
- Inefficient overlay routing
  - Potential to reduce network costs a lot
  - Instead, today's P2P increases them
- More transit costs

RTT in seconds, cause

- The uplink is 250-500kb/s
- The buffer is 32-64kB
- No AQM
- TCP fills the buffer

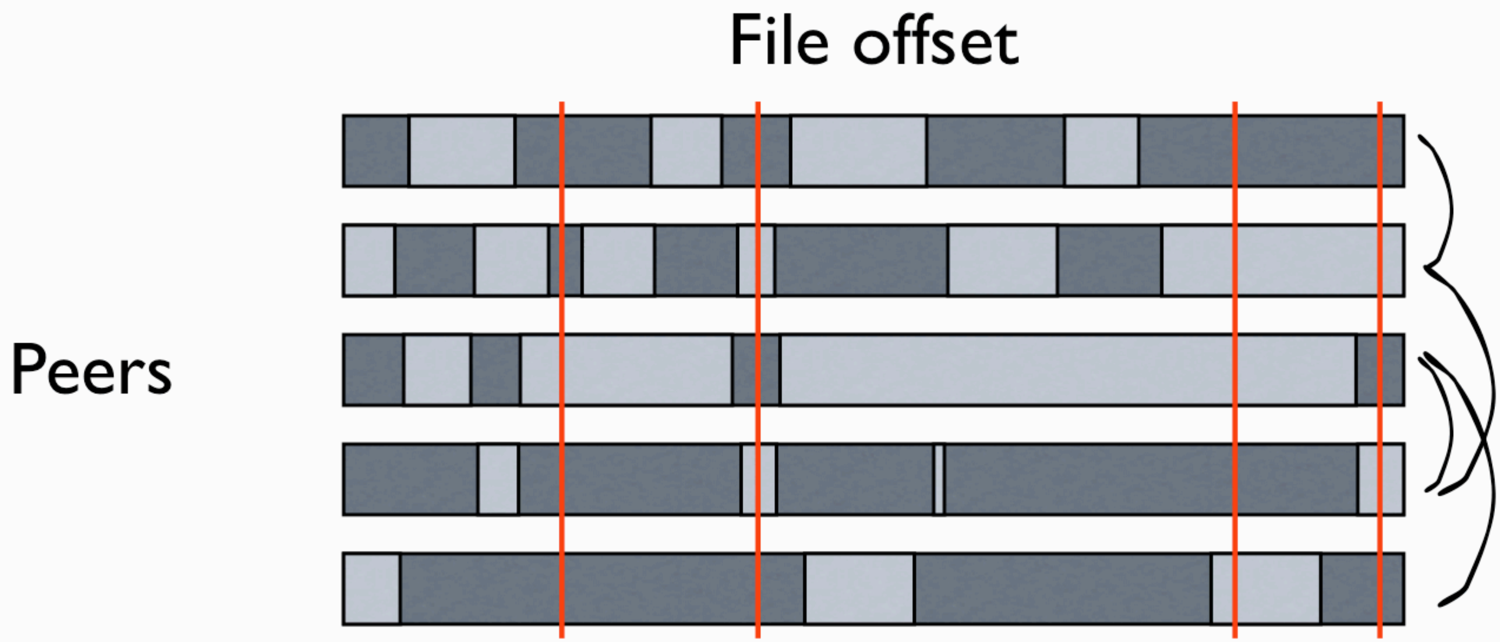
RTT in seconds, effect

- The network is “slow”
- Web browsing barely possible
- VoIP and games essentially unusable
- Might affect neighborhood on cable?

Random peer selection

- BitTorrent gets a list of peers from tracker
- Tracker randomizes the list
- Client chooses randomly
- A bit of bias towards established peers
- Otherwise **random**

Rarest first, Piece dissipation



Why random peers?

- Rarest-first ensures best reliability
- Rarest-first helps to have pieces to trade
- Global rarest-first is approximated by local
- Best approximation if sample is unbiased
- Peer diversity
- Best peers can be far away

Solving RTT in seconds

- Different congestion control
- Scavenger service
- More explicit congestion notification

Making overlay efficient

- Tradeoff with rarest-first, but it's OK
- Prefer peers in same AS — simplest
- Prefer peers in set of ASes — need the set
- Prefer peers in list of IP prefixes — same
- A cost minimization algorithm

How much does smart peer selection win?

- Case study: four most popular torrents
- AS with >20 peers could avoid transit
- Swarm sizes: 9984, 3944, 2561, 2023
- Peers in ASes with >20 peers: 6863, 1926, 1045, 673
- 57% are in ASes with >20 peers
- Could reduce transit traffic at least by 57%

Caching

- What if there aren't enough local peers?
- Make one then, a fast one. Or 10, or 100.
- Bonus: reduces uplink use on last-mile

How large does the cache need to be?

- For 30% hit rate, 1 TB
- For 80% hit rate, 100 TB
- 1 TB fits into a device
- 100 TB can be assembled from 100 devices

Congestion control design goals

- Keep bottleneck full
- Keep delay lower than unloaded + ε
- Yield to TCP on forward path
- Separate reverse-path congestion
- React in 1 RTT

Congestion control approach

- Continuously estimate one-way delay
- Separate into propagation and queuing
- Target a small value for queuing

Congestion control status

- Implemented
- Instrumented
- Tested in the lab
- Tested on the Internet with 7M users
- Works as designed
- Further reductions in extra delay in future

Scavenger service

- Mark traffic with a given DSCP (001000?)
- WFQ scavenger class into a 1% allocation
- Make the buffer short in scavenger queue
- Only helps where you can tweak the router
- Probably does not help at the last mile

More explicit congestion notification

- Learn about queue before FIFO drops
- AQM+ECN would be an improvement
- Better yet, tell the ends the queue size
- Would aid all kinds of congestion control
- **Not a short-term solution**

Best practical solutions

- Caching
- Smart peer selection
- Better congestion control

IETF role

- Cache discovery protocol
- Net info for smart peer selection
- Experimental congestion control
- BCP on P2P pain

BitTorrent cache discovery protocol

- BEP 22, bittorrent.org/beps/bep_0022.html
- DNS-based
- SRV_bittorrent_tracker(rDNS(external IP))
- Remove left component of domain name until a hit

Net info for smart peer selection

- We're adding preferring local AS. Is it useful?
- Would like to select peers in way that minimizes ISP costs
- Need more information about costs
- Expose some BGP information so the overlay can be no worse than underlying?

Experimental congestion control

- A framework for congestion control with design goals different from TCP's
- Our congestion control is specific for µTorrent
- Other apps could benefit

Next steps

- Cache discovery
- Net info for smart peer selection