Cooperative Browser Download Streams

- Overview of Presentation
  - Brief review of the problem, the proposed solution, and why it's interesting and hard
  - Does the proposed solution have potential?
    - Look at the results of simulation using Kazaa trace
  - The implementation
  - Previous work
  - Future work

Cooperative Browser Download Streams

- The Problem
  - Users want to download content as fast as possible
  - Servers want to serve content as efficiently as possible
  - This becomes harder with higher-bandwidth content, and as the load on servers increase

Cooperative Browser Download Streams

- Why is it interesting?
  - Servers can't keep up with demand
  - They need to use things like Content Distribution Networks
  - This costs $$$
  - Is there any way to solve the problem without the service provider incurring additional costs?

Cooperative Browser Download Streams

- The Solution
  - Users who are downloading from the server should share the content that they are downloading
  - Other users don't have to go to the origin server to get the content
  - Load is reduced on the server
  - Service providers don't have to spend extra $$$ – onus is on end-users to share the content

Cooperative Browser Download Streams

- Why is it Hard?
  - Incentive
  - Transparency
  - Privacy

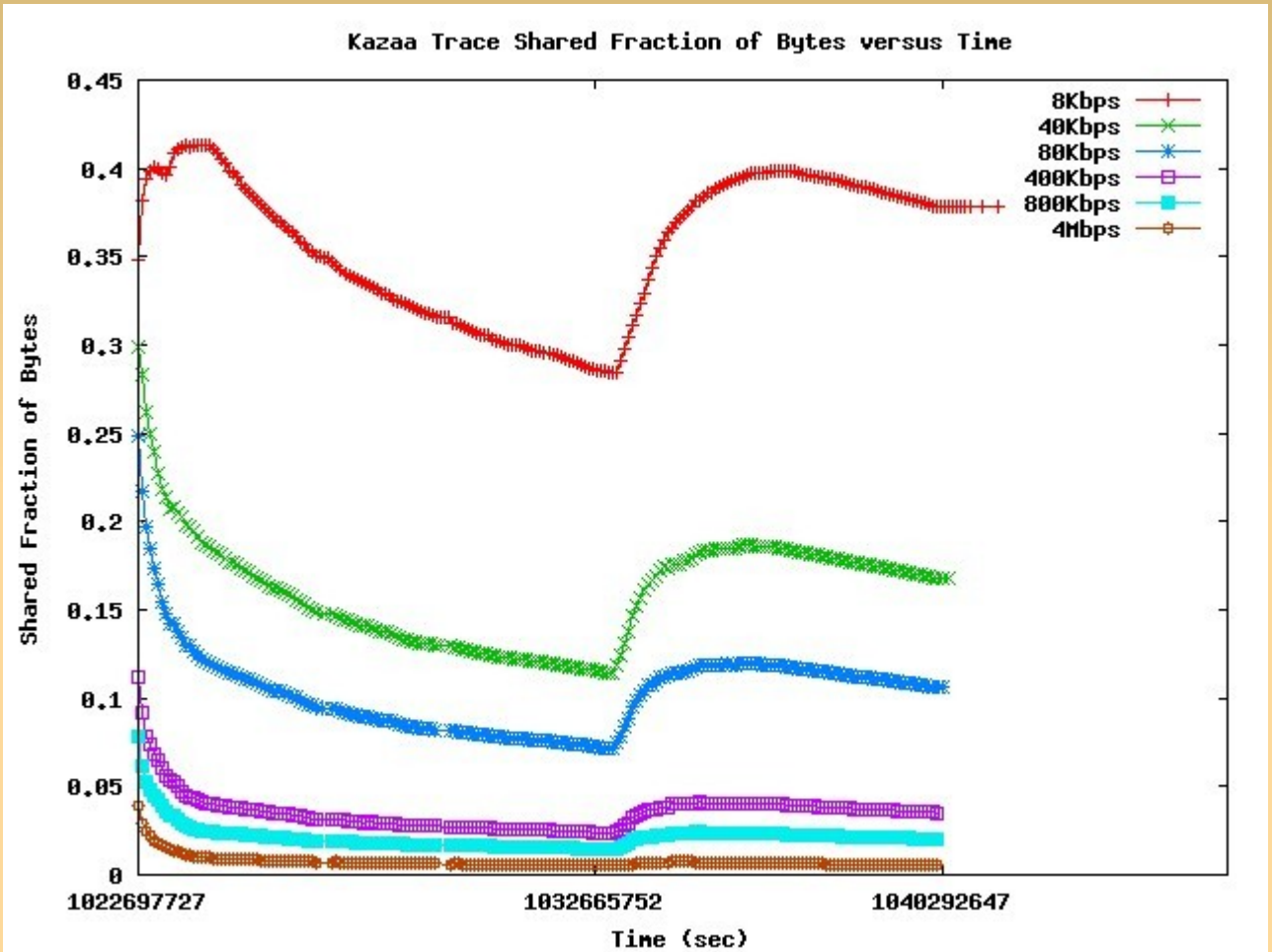
Cooperative Browser Download Streams

- Does the solution have potential?
  - Simulations that I ran ask:
    - What fraction of the total bytes being downloaded can be served by peer nodes instead of the origin server? (i.e., potential server bandwidth savings)
    - What is the number of concurrent uploads that a client potentially has to serve?

Cooperative Browser Download Streams

- Does the solution have potential?
  - Simulations use Kazaa trace of downloads spanning 200 days
    - Pruned original trace file to contain only files whose size was >= 100M
  - Idea is to simulate “aggressive sharing” -- what is the upper limit on bandwidth savings?
  - Assume a number of different possible transfer rates

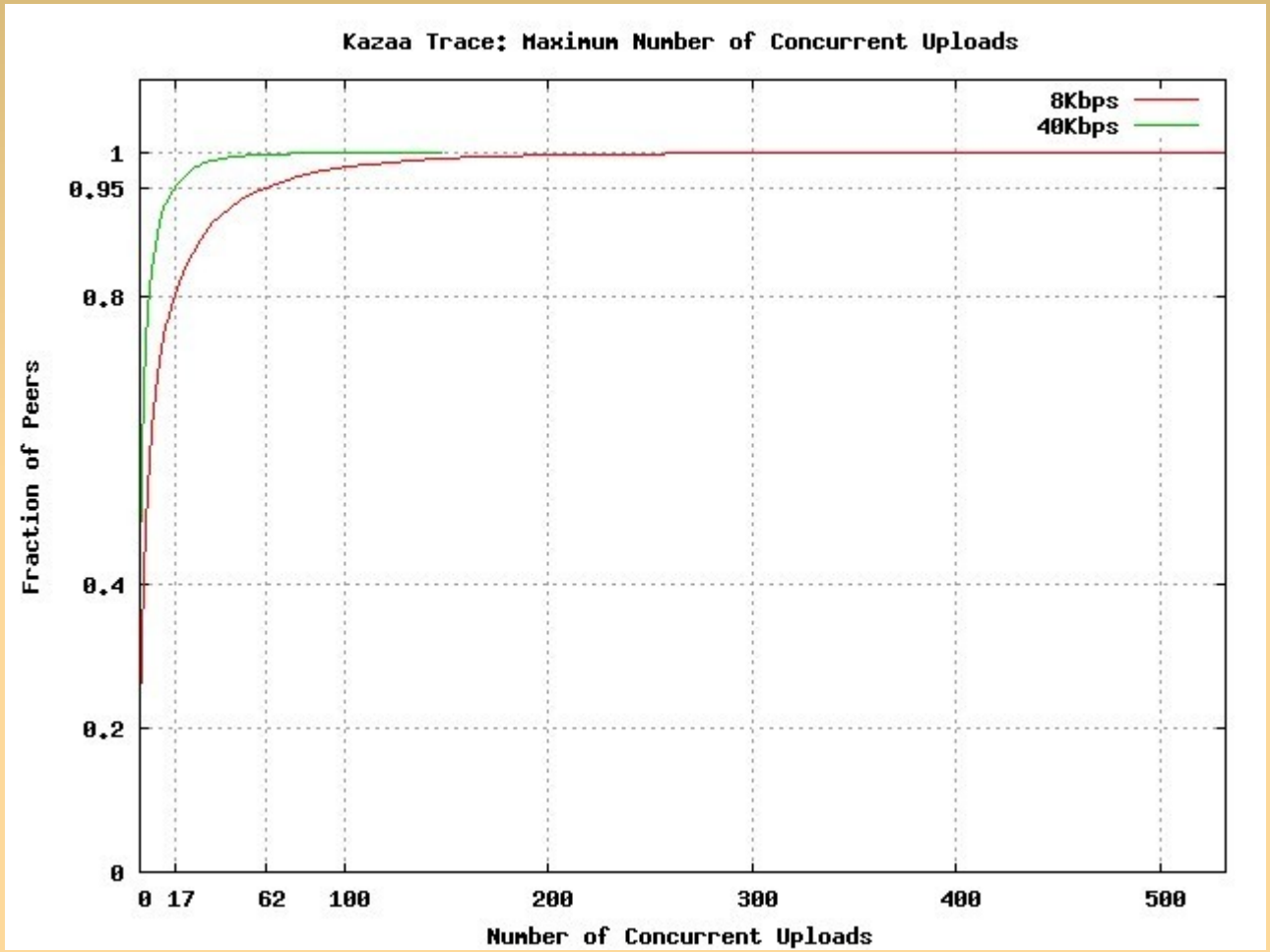
Cooperative Browser Download Streams



Cooperative Browser Download Streams

- Fraction of bytes shared
  - 8Kbps (1K bytes per second) – 37.8%
  - 40Kbps (5K bytes per second) – 16.8%
  - 80Kbps (10K bytes per second) – 10.6%
  - 400Kbps (50K bytes per second) – 3.5%
  - 800Kbps (100K bytes per second) – 2.0%
  - 4Mbps (500K bytes per second) - 0.55%

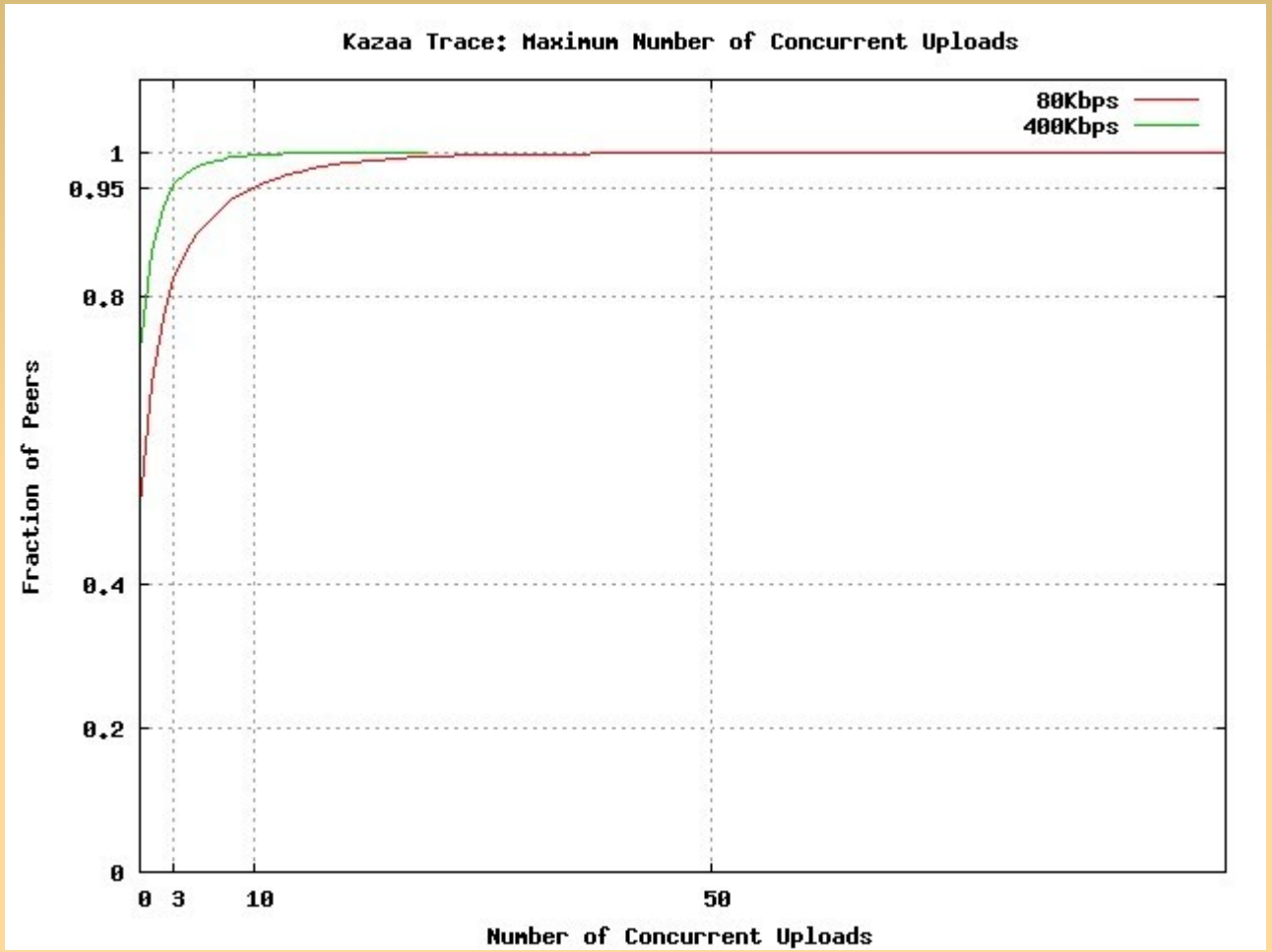
Cooperative Browser Download Streams



Cooperative Browser Download Streams

- 8Kbps – At most 62 concurrent uploads by 95% of peers. Some peer serves 532 clients.
- 40Kbps – At most 17 concurrent uploads by 95% of peers. Some peer serves 148 clients.

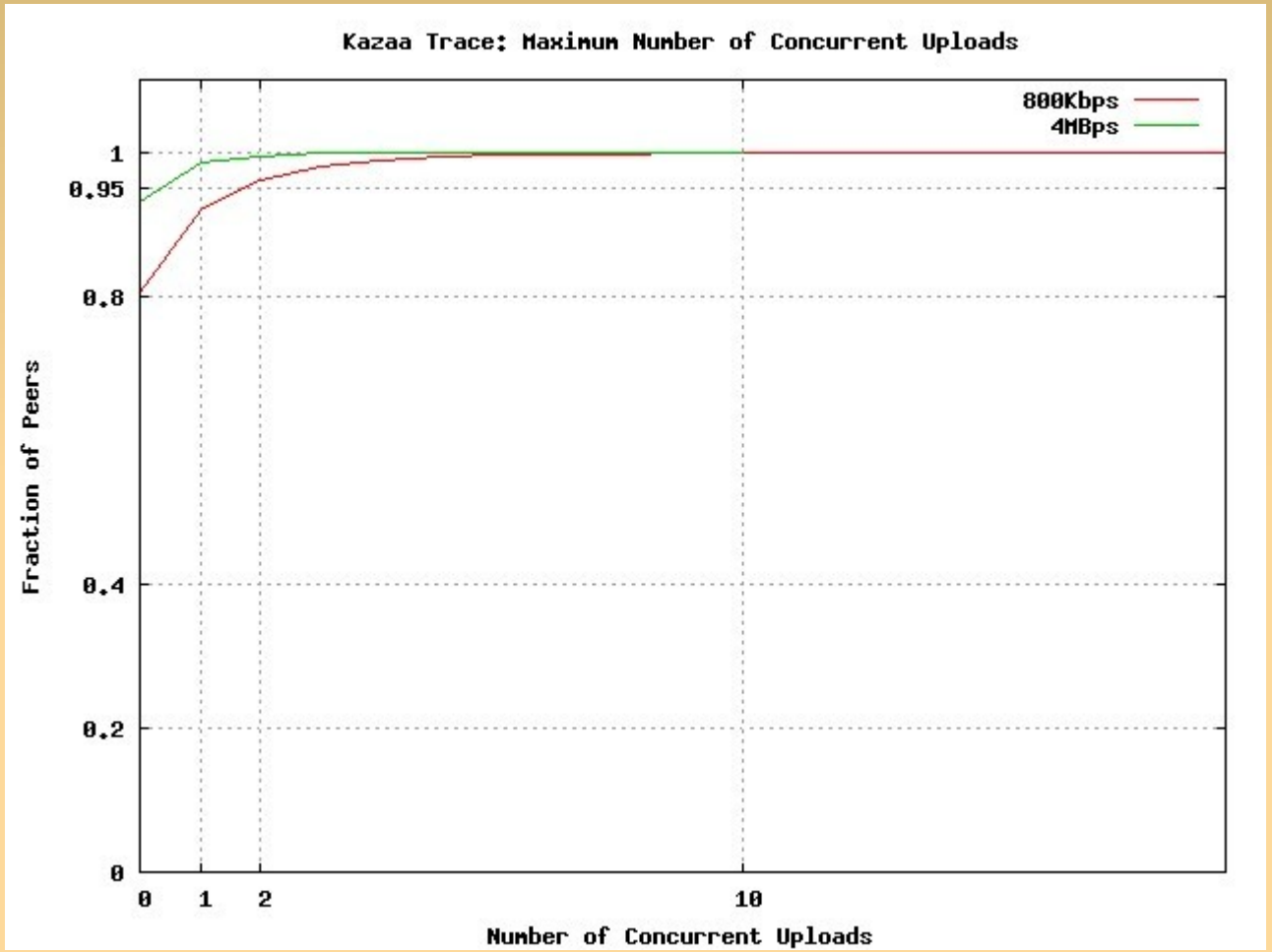
Cooperative Browser Download Streams



Cooperative Browser Download Streams

- 80Kbps – At most 10 concurrent uploads by 95% of peers. Some peer serves 95 clients.
- 400Kbps – At most 3 concurrent uploads by 95% of peers. Some peer serves 25 clients.

Cooperative Browser Download Streams



Cooperative Browser Download Streams

- 800Kbps – At most 2 concurrent uploads by 95% of peers. Some peer serves 18 clients.
- 4Mbps – At most 1 concurrent uploads by 95% of peers. Some peer serves 10 clients.

Cooperative Browser Download Streams

- Implementation
  - Firefox extension bootstrapping a proxy written in Java
    - Proxy runs in its own threads in the Firefox browser.
    - User does not have to configure proxy settings.
    - Transparency

Cooperative Browser Download Streams

- Implementation
  - Idea: Use OpenDHT to index clients available for sharing download streams
  - When making HTTP requests for static content, do a lookup in OpenDHT (via XML RPC) to see if any clients can serve the content.
  - When receiving HTTP response for static content, register yourself in OpenDHT as a client available for serving that content.

Cooperative Browser Download Streams

- Implementation
  - Use HTTP range requests to obtain part of file not fully downloaded from a peer

Cooperative Browser Download Streams

- Problems encountered
  - OpenDHT can be slow
    - Recent improvements
  - Some servers don't follow the HTTP RFC (2616).
    - slashdot.org is an example. Must use absolute path instead of absolute URI or you get 503 Service Unavailable error
    - Proxy needs to be tolerant of this

Cooperative Browser Download Streams

- Previous Work
  - On the scale and performance of cooperative Web proxy Caching. Wolman, et al.
  - Squirrel: A decentralized peer-to-peer web cache. Iyer, et al.
  - The Case for Cooperative Networking. (Microsoft CoopNet Project) Padmanabhan, et al.

Cooperative Browser Download Streams

- Future Work
  - How to handle Firewalls & NATs
  - Add support for HTTP 1.0
    - Implementation uses chunked transfer coding for transmitting data between peers.
  - Privacy issues
  - Server cooperation for providing incentive for users to participate