Advanced Topics in Computer Systems, CS262B
Prof Eric A. Brewer

P2P Measurement, Modeling and Analysis

March 16, 2004

I. Background

P2P traffic 43% of bandwidth at UW, vs. 14% for web traffic

- multimedia files are immutable and large
- relatively few distinct documents

II. The Trace

Details:

- 203 days
- 1.6M requests (Kazaa)
- 98M transactions of which 66% fail
- 25K users (all within UW) (out of a population of 60K)
- 600K unique objects totaling 9TB
- Possibly 60% of all movies released?

Only looks at UW internal users getting data from *external* servers

Representative? Biases?

Note the at-collection-time anonymization

No Kazaa control traffic; removed auto-upgrade sessions

III. Users

Patient:

- small: 30%> 1 hour, 10%> 1 day
- large: 50%> 1 day, 20%>1 week

Not interactive: batch-mode delivery

Experienced users still use the system, but ask for smaller objects

There is some attrition

Client activity: how often are clients active?

- hard to measure availability here

- session length only 2.4 minutes!
  - most xacts fail, client may be idle for a while looking for an available server, some xacts are short.

IV. Objects

(these) files are immutable

Clients fetch and object at most once

- 94% at most once, 99% at most twice, 1% more than twice (?)
- web driven by changing documents (Yahoo, CNN, Google)

Popular objects change quickly over time, more so for audio than video

- most popular objects are new (newly born)
- but most traffic goes to older objects
- video objects change popularity more slowly

Zipf Distributions:

- Zipf's law: the popularity of the i^{th} most popular object is $i^{-\alpha}$
- linear on a log scale (down and to the right)
- George Zipf was a linguist. He showed that the usage of English words follows this distribution, as does city population.
- Kazaa not Zipf *because* no reason to re-request objects -- the most popular objects are that popular
- But still enough difference in popularity to motivate caching
- Nice trick: simulate Zipf then eliminate multiple requests to the same object => looks like Kazaa curve
- Video-on-demand was thought to be Zipf, but is not -- nice use of log-log plots!
- The real issue: the objects themselves are very dynamically -- what does it mean to have a Zipf distribution if objects come and go and the popularity changes all of the time? All we can say is that a single snapshot has some popularity distribution (which is not Zipf).

V. Caching

Hit rate should decrease over time with a stable population of objects (popular objects get hit early)

But new arrivals increase the hit rate!

New clients could keep the hit rate up too, but this would be a "pyramid scheme" -- each new client has to to lead to more new clients in the future.

Perfect proxy cache would get 86% hit rate, which is great!

- but Kazaa is not locality aware...

Solution 1: centralized request redirector + indexer (mini-Napster)

Solution 2: decentralized request redirection teach supernodes about locality

Solution 3: Eric's solution: one big supernode on the inside that stores everything

In general, 60-80% savings of outgoing bandwidth... even though most nodes are considered "down" unless they are active (very conservative). Reason: a few highly available servers serve most content...

HA peers are necessary and sufficient for caching... (and easier to build). They also have the most files.

- these peers seem to have most files already (which means they are essentially Napster-like centralized servers with the content as well as the index!)
- these are the nodes that RIAA is targeting -- they are also easy to find!

Peer availability does not matter as much as object availability

- also hard to measure -- what happens when a node leaves permanently? when does it stop counting as unavailable
- some systems trick their users into leaving the client running, which inflates the peer availability

What about multicast or broadcast?

What about eviction policies?

Supernodes could collaborate to increase effective cache size...

Can you use a search engine to find the files instead? (it is a better centralized index)