Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts

S. Saroiu, P. Gummadi, and S. Gribble

Multimedia Systems Journal
Volume 8, Issue 5

November 2002

Introduction (1 of 2)

- Peer-to-Peer (P2P) file sharing have created interest in P2P architectures
- Exact definition debatable, but P2P
  - Lacks centralized infrastructure
  - Depends upon voluntary participation for resources
- Membership ad-hoc and dynamic
  - Capacity, latency, availability of peers change

  → Must be aware of when deciding suitable peer for allocating resources

Introduction (2 of 2)

- However, few architectures are evaluated considering suitability of peers
  - Due do lack of characteristics on hosts
- This paper
  - Studies Napster and Gnutella (were the two most popular)
  - Seeks to precisely characterize the population of end-user hosts
    - Typically home machines on the "edge" of the Internet

http://www.slyck.com/

This Paper

- Characterization
  - Bottleneck capacities
  - Latencies
  - Availability
  - Number of files
  - Correlations between above stats
- Lessons
  - Heterogeneity – 3-5 orders of magnitude
  - Peers deliberately mis-report information if they have incentive to do so. Need:
    - Built-in incentives to tell the truth
    - Ability for system to verify peer information

Outline

- Introduction (done)
- Methodology (next)
- Results
- Recommendations
- Conclusions

Measurement Methodology

- Periodically crawl each system
  - Gather snapshots:
    - IP and port and reported information
    - Do some active measurements
- Sub-sections
  - Architectures
  - Crawling
  - Active Measurements
  - Limitations

Napster and Gnutella Architectures

- Peers function as client server
- Query for file, download from peer

- Napster has centralized server index
  - Servers keep track of peer information
- Gnutella has overlay network
  - floods requests (TTL to limit scope)
  - ping and pong messages to discover peers

The Napster Crawler

- Server architecture
  - ~160 Napster servers, peers connect to 1
  - Server reports "local" and "remote" users
- Actively query popular song artists, see what peers responded (do in parallel, so only takes 3-4 min.)
  - By comparing to global server stats, captured 40-60% of peers with 80-90% of traffic
  - Distribution of remainder traffic stats similar
- For each peer discovered, request
  - Capacity of peer as reported by peer
  - Number of files being shared
  - Number of uploads and downloads in progress
  - Names and sizes of files
  - IP address of peer

The Gnutella Crawler (1 of 2)

- Connect to several well-known peers
  - gnutellahosts.com, router.limewire.com
- Send ping messages with large TTL
- Add new peers based on pong messages
  - Gives IP, number and total size of files
- Should be no bias since not using "popular" songs
- Allow ~2 minutes, report peers
  - Usually, about 8000-10000 hosts

The Gnutella Crawler (2 of 2)

- Based on clip2 (gnutella measurement)
  - about 25-50% of hosts at that ime

Measurement Methodology

- Periodically crawl each system
  - Gather snapshots:
    - IP and port and reported information
    - Do some active measurements
- Sub-sections
  - Architectures
  - Crawling
  - Active Measurements
  - Limitations

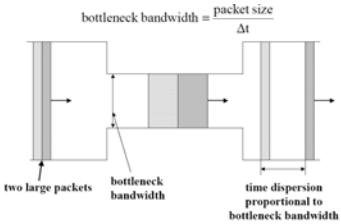
Active Measurements

- P2P sys only report limited info (and sometimes not accurate) about peers
  - Peers may choose not to report capacity
  - Peers may lie to discourage downloads
- For each snapshot, gather direct data
  - Capacity, latency, num files, lifetime
- Next, discuss:
  - Bottleneck capacity measurements
  - Latency measurements
  - Lifetime Measurements

Bottleneck Capacity Measurements

- Real number would be available capacity
  - But would require TCP connection, so costly
- Instead, try to measure maximum capacity
  - An approximation of available
  - Report bottleneck (lowest) capacity
- Existing techniques (flood one packet, or several packet-pairs) not acceptable
  - "flood" causes too much traffic
  - Several packet pairs can not take 1 minute, so 1 week for 10k measurements
  - Cannot deploy custom software on all hosts
  - → Design SProbe

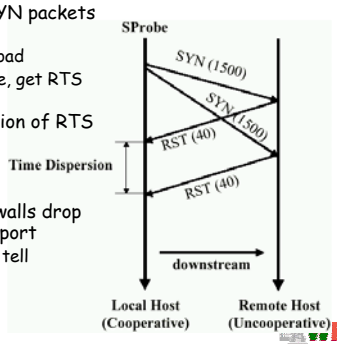
SProbe (1 of 4)



- Dispersion of two large packets gives measure of bottleneck
  - The larger, the slower the link
- How to get peers to report? Rely upon response

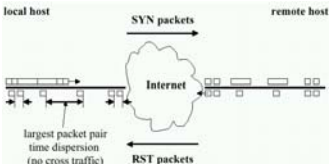
SProbe (2 of 4)

- Send two TCP SYN packets back-to-back
  - Add large payload
  - If port inactive, get RTS packet back
- Measure dispersion of RTS packets
- Note, some firewalls drop SYN to inactive port
  - SProbe cannot tell difference



SProbe (3 of 4)

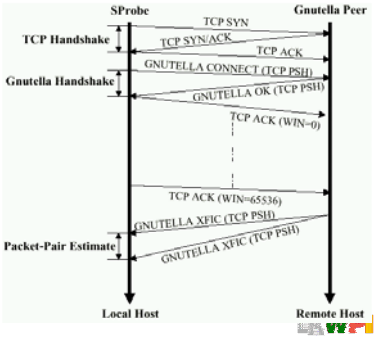
- Cross traffic can interfere with dispersion
  - Current approaches send lots, but doesn't scale and takes too long
- Send packet train, small at ends, large in middle



- If dispersion of small is larger than large, assume there may be cross traffic and return "unknown"

SProbe (4 of 4)

- For upstream, need peer to send
- Initiate Gnutella handshake
- Wait so build up large packets
- When send, measure dispersion



Latency Measurements

- TCP throughput directly dependent upon latency (round-trip time)
  - $T = k / [RTT \times \text{sqrt}(p)]$
- Measure time for 40 byte TCP packet exchange (minimize bottleneck transmission)
- While P2P may be different than P2Server, distribution to well-connected server still of interest

Lifetime Measurements

- States:
  - *Offline* – not connected or behind firewall
  - *Inactive* – connected but not doing P2P
  - *Active* – participating in P2P
- Send TCP SYN to P2P port
  - If no packet, then offline
  - If RST then inactive
  - If SYN/ACK then active

Summary of Active Measurements

- Lifetime - random subset of peers
  - 17,125 Gnutella peers over 60 hours, every 7 minutes
  - 7,000 Napster peers over 25 hours, every 2 minutes
- Bottleneck and Latency
  - Tried 595,974 Gnutella peers, only 223,552 reliable downstream, 16,252 upstream, 339,502 latency
  - Tried 4079 Napster peers, with 2049 successful (complaints of "intrusive" forced to stop early)

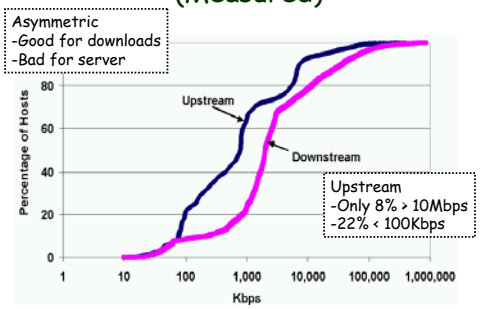
Limitations of Methodology

- Ideal should include workload
  - So could see how to tune system
- Ideal should know "birth" rate so know how will scale
- May incorrectly classify peers
  - IP addresses may be shared (multiple hosts behind NAT box), but think are one
  - IP addresses may be re-used (DHCP), so "same" peer moves
- Little (scientific) knowledge about broadband so unclear of effects on performance
  - Packet loss, congestion … (queues! ☺)

Outline

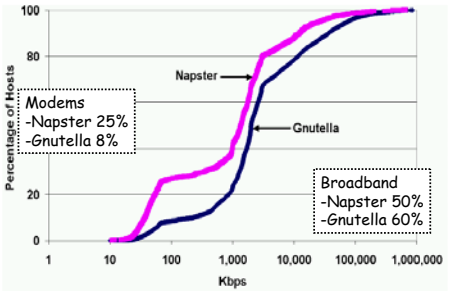
- Introduction (done)
- Methodology (done)
- Results (next)
- Recommendations
- Conclusions

Peers with Server-Like Capacity? (Measured)



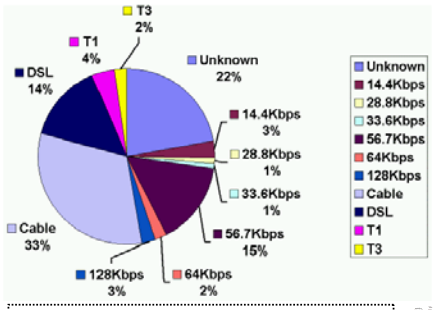
Gnutella Peers

Measured Download Capacity



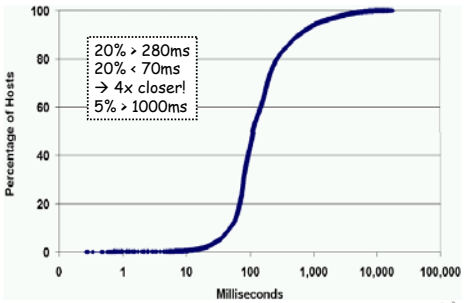
-Gnutella needs flooding, so more capacity
-Gnutella rumor is more technical, and they have more capacity

Reported Capacities for Napster



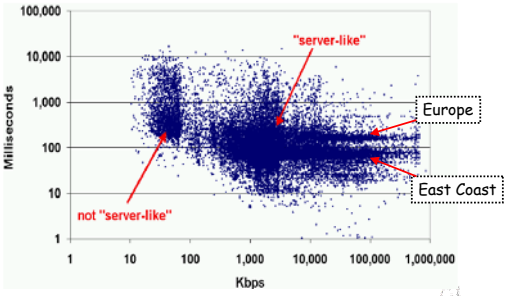
Unknowns may be mis-reporting to avoid downloads (MLC: maybe they don't know? Other ratios match)

Measured Latencies for Gnutella

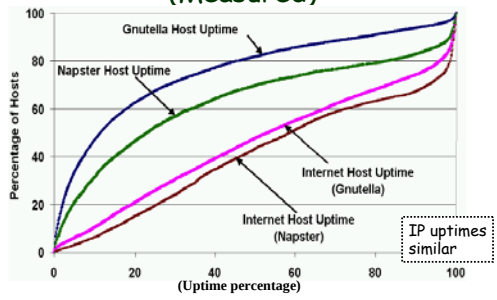


20% > 280ms
20% < 70ms
→ 4x closer!
5% > 1000ms

Server Like Gnutella Peers?



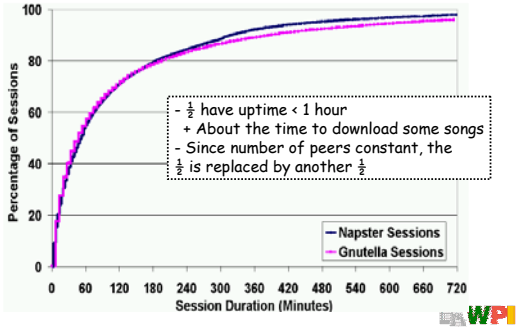
Peers with Server-Like Uptimes? (Measured)



IP uptimes similar

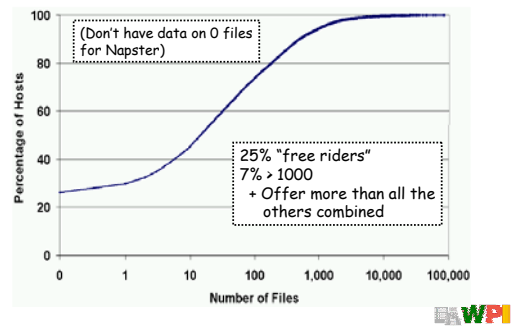
Napster peers participate more. Perhaps due to "chat" and "MP3" in Napster? Or maybe more useful?

Session Durations



- ½ have uptime < 1 hour
  + About the time to download some songs
- Since number of peers constant, the ½ is replaced by another ½

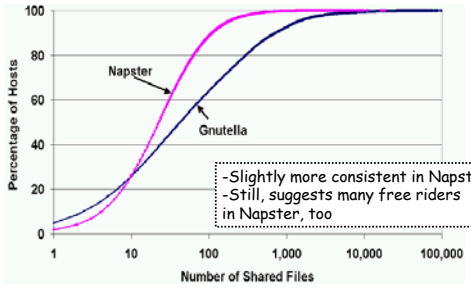
Number of Shared Files - Gnutella



(Don't have data on 0 files for Napster)

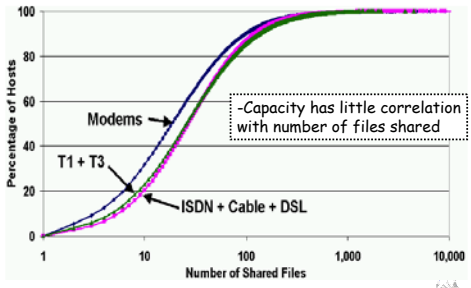
25% "free riders"
7% > 1000
+ Offer more than all the others combined

Number of Shared Files ("Zero" Files Removed)



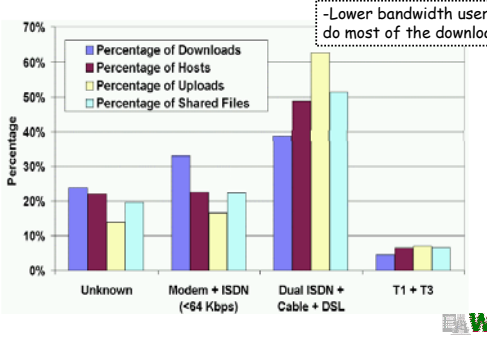
-Slightly more consistent in Napster
-Still, suggests many free riders in Napster, too

Number of Shared Files – Napster (Reported Capacity)



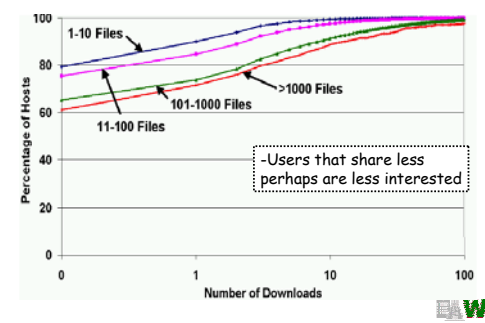
-Capacity has little correlation with number of files shared

Number of Downloads



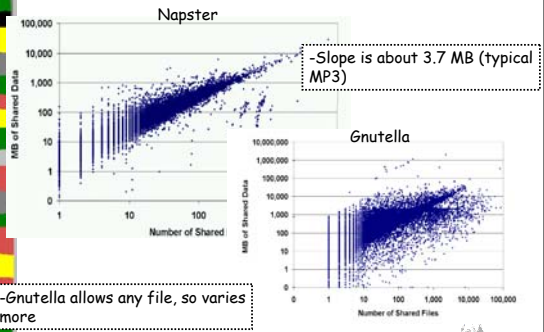
-Lower bandwidth users do most of the downloads?

Downloads versus Sharing - Napster



-Users that share less perhaps are less interested

Shared Files versus Total Size

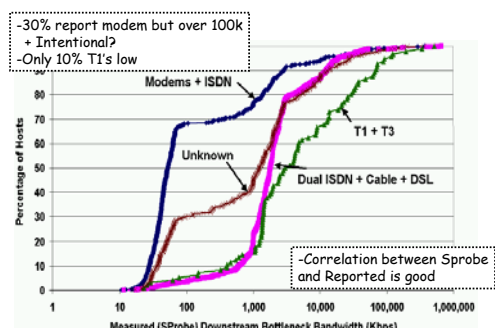


-Slope is about 3.7 MB (typical MP3)

-Gnutella allows any file, so varies more

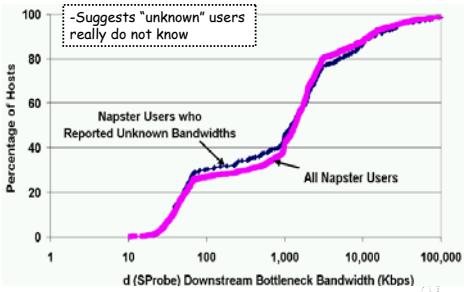
Measured Downstream Bottleneck

-30% report modem but over 100k + Intentional?
-Only 10% T1's low

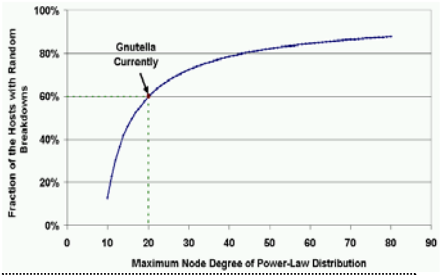


-Correlation between Sprobe and Reported is good

Measured Downstream Bottleneck



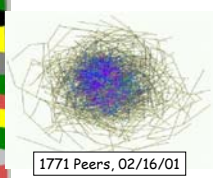
Resiliency in the Face of Failure



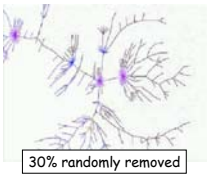
- Predicted resiliency based on measured connectivity
- Uses power-law that is based on degree of connectivity
  + But nodes "prefer" good nodes so some well-connected

-But what if a "malicious" attack at best connected nodes?

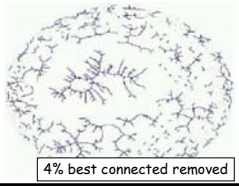
Gnutella Topologies



1771 Peers, 02/16/01



30% randomly removed



4% best connected removed

-Malicious, well-placed attack can shatter even reslient P2P networks!

Recommendations

- P2P systems assume equal peers
  - But extreme heterogeneity across capacity, latencies, lifetimes and shared data
  - → Instead, should delegate across hosts based on physical characteristics
- P2P systems assume equal participation
  - But clearly some download most, serve least
  - → Maybe impose equality if want equal performance
- P2P systems assume users want to cooperate
  - But users will misrepresent if it gives them advantage
  - → Instead, should try to measure instead of trusting

Conclusions

- Measured popular P2P file sharing systems with many voluntary users
- Lessons:
  - Significant heterogeneity
  - Clear asymmetric behavior in users
  - Peers deliberately mis-report information if it helps them to do so

Future Work (MLC)

- Other systems: KaZaa, E-Donkey, BitTorrent…
- New P2P systems based on lessons from this paper
  - Delegate based on capabilities, for example
- Measure total downloads, characteristics of content