
The Scalability of Swarming Peer-to-Peer Content Delivery

Daniel Zappala
Brigham Young University
zappala@cs.byu.edu

with

Daniel Stutzbach
Reza Rejaie
University of Oregon

Motivation

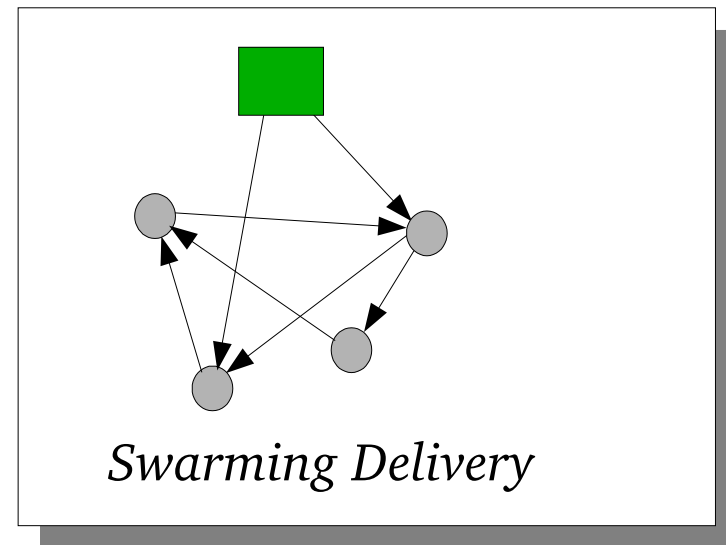
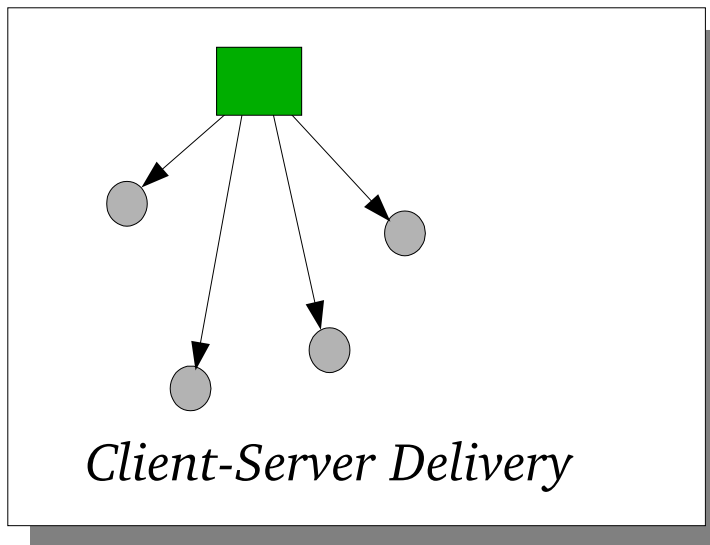
- Small web sites can't handle high loads (large files or large audiences)
 - Client-server is an inherent bottleneck
 - As load increases, server must either refuse new clients (to preserve good performance for current clients) or allow delay to increase
 - Can't predict popularity – *flash crowds* occur when a site suddenly becomes popular

Existing Solutions

- Buy more bandwidth – expensive and ineffective (bottleneck remains)
- Content Distribution Network (CDN)
 - Distribute content to high-bandwidth servers
 - Redirect clients to nearest server
 - Expensive and you must know ahead of time that you will have popular content
- Caching – helps some clients, must be universally deployed to help the server
- Multicast – heterogeneity, timing, lack of deployment

Swarming: Peer-to-Peer Content Delivery

- *Peer-to-Peer Networking: All participating hosts can both provide and receive service*
- As applied to swarming
 - any client that downloads any portion of a file can re-serve that file to other clients
 - Clients can download pieces of a file in parallel from their peers



Swarming Advantages

- Better performance for users – shorter download time
- Server can handle higher load
 - Uses scale to its advantage – system capacity increases with the number of peers in the system
- Avoids network congestion – load imposed by content delivery is spread over the entire network
- Parallel download protects against peer bottlenecks and peer instability
- Inexpensive – takes advantage of unused client bandwidth (like [SETI@home](#) uses idle CPU cycles)

Existing Systems

- BitTorrent is regularly used to transfer new software releases to hundreds of peers
 - Several other proprietary and open-source implementations
 - This is becoming the norm with peer-to-peer networking – implementation precedes research
- We know that it works, we don't know how well it works and if it can do better
- Existing measurement studies don't tell us how well it can scale

Complexity of the System

- Many dynamics involved, even in a basic system
 - Finding peers with the desired content
 - Choosing peers that are likely to provide good performance
 - Managing parallel download while coping with
 - Partially available content at each peer
 - Changes in available bandwidth from each peer
 - Peer instability
- Almost makes TCP look easy – we're still trying to get that right

Our Contributions

- Examination of the scalability of peer-to-peer content delivery
 - Developed a swarming architecture – key design components
 - Simulation-based evaluation
- Findings
 - Swarming can scale with offered load several orders of magnitude beyond what a typical web server can handle
 - Swarming can smoothly handle flash crowds, with minimal effect on client performance
 - Swarming can scale to a wide range of file sizes, block sizes, and client bandwidths

Related Work

- BitTorrent

<http://bitconjuror.org/BitTorrent>

- Centralized tracker coordinates peers
- Incentives built in – serve data to peers that serve you

- CoopNet and Pseudoserving

V.N. Padmanabhan and K. Sripanidkulchai, The Case for Cooperative Networking, IPTPS, 2002.

K. Kong and D. Ghosal, Mitigating Server-Side Congestion on the Internet Through Pseudo-Serving, IEEE/ACM ToN, August 1999.

- Server gives client a list of peers, client chooses one
- Clients are able to find content, load is balanced
- Vulnerable to client bottlenecks and client instability

Related Work

- Backslash

T. Stading, P. Maniatis, and M. Baker, Peer-to-Peer Caching Schemes to Address Flash Crowds, IPTPS, 2002.

- Collaborative web mirrors
- Limited scalability – clients far outweigh servers

- PROOFS (ICNP, 2002)

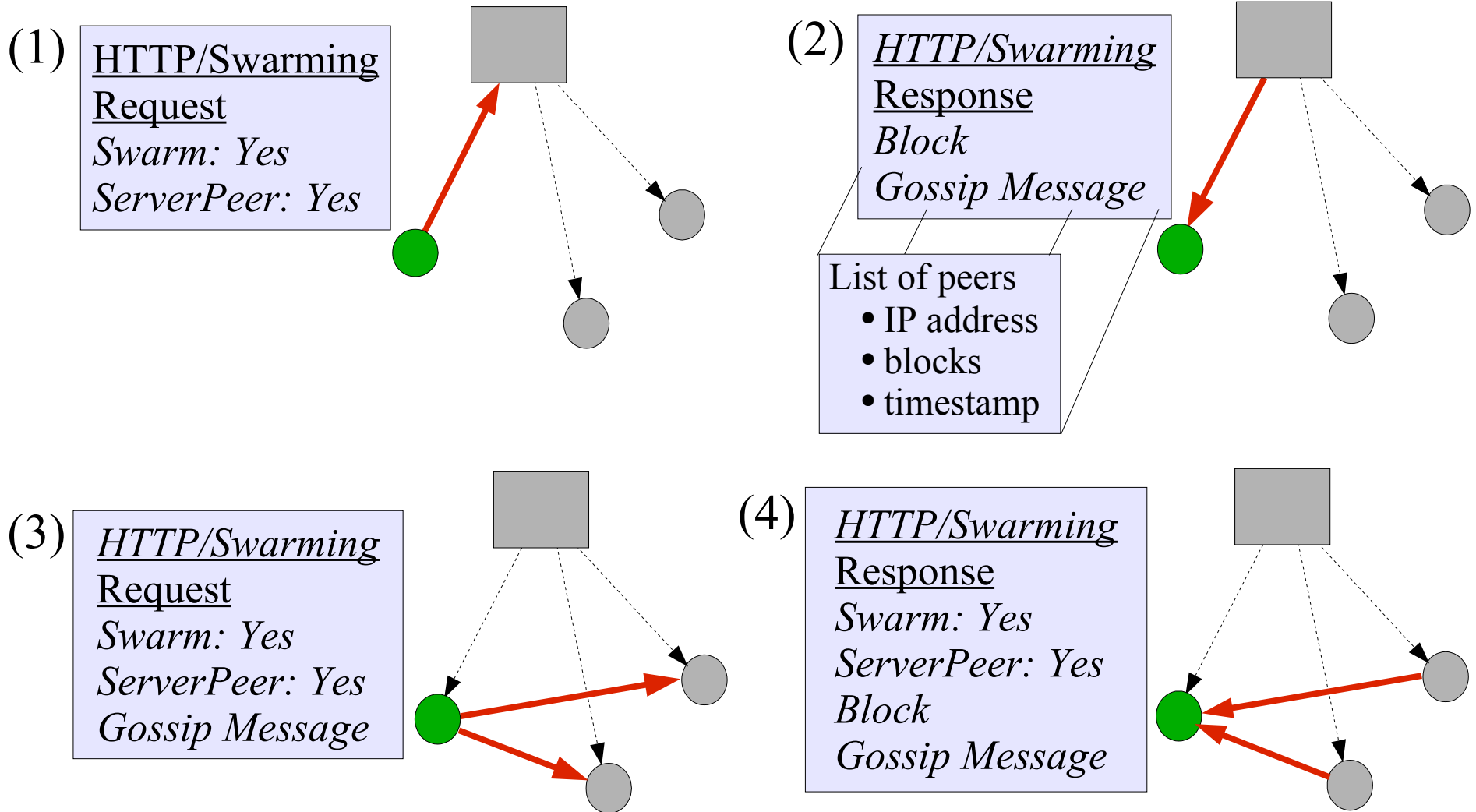
A. Stavrou, D. Rubenstein, and S. Sahu, A Lightweight, Robust P2P System to Handle Flash Crowds, IEEE ICNP, November 2002.

- Consult P2P network after web server is overloaded
- Complementary to swarming, but doesn't prevent overload

Swarming Architecture

- Four components:
 - swarming initiation: decide when to use swarming versus client-server delivery
 - peer identification: find peers with the desired content
 - peer selection: choose a peer to deliver each piece
 - parallel delivery: download from selected peers
- Hybrid between client-server and peer-to-peer
 - Client-server for small or unpopular files, to bootstrap peer identification, fallback in case peers leave
 - Peer-to-peer for scalable delivery, peer identification and selection

Swarming Overview



Swarming Initiation

- When should the server start or stop swarming for a particular file?
- What should the server give a client? The entire file? Only one block? Just a gossip message?
- Our approach: be conservative
 - Swarm at all times
 - Avoid reacting too slow to sudden increase in load
 - Increased delay during low load (peers slower than server)
 - Give at least one block to each client
 - Balance between giving too little and too much
 - Force clients to cooperate

Peer Identification

- How does a client find peers with the desired content?
 - Not known ahead of time, highly dynamic
 - Fortunately, a client only needs some peers, not all
- Our approach: server + gossiping
 - Server supplies an initial set – simple bootstrapping
 - Clients/peers exchange gossip messages
 - Peers, the blocks they hold, and a timestamp
 - Freshness is more important than content
 - What good is a peer who has everything but has left?
 - Be sure to gossip about peers who have left
 - Each client caches N_c peers with most recent timestamps, gossips about most recent N_g , $N_g \leq N_c$

Peer Selection

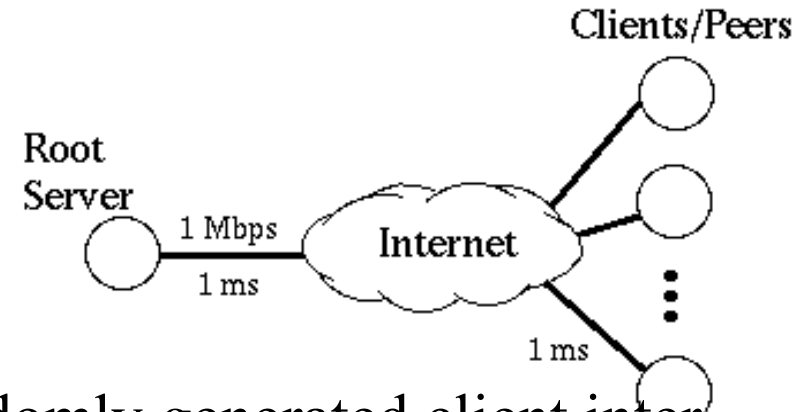
- Which peers and how many peers should a client use for parallel download?
 - Available bandwidth not known ahead of time
 - Bottleneck may be remote or local
- Our approach: content availability
 - Don't try for optimal approach yet – not sure how much it affects performance
 - Content availability more important than performance
 - What good is a fast peer without anything you need?
 - Each client limits itself to N_d concurrent downloads
 - Choose a peer with the most blocks the client needs

Parallel Delivery

- When should the client add or drop a peer? Which blocks should the client download from each peer?
- Our approach: simplicity and stability
 - Each client chooses N_d peers using peer selection
 - Keep using this set unless a peer disconnects or doesn't have any blocks the client needs
 - Return to server if unable to find any useful peers
- No performance monitoring of peers
 - May lead to instability
 - Faster peers will naturally deliver more blocks
- Server – large block size to keep peers busy, choose initial blocks randomly to ensure diverse content

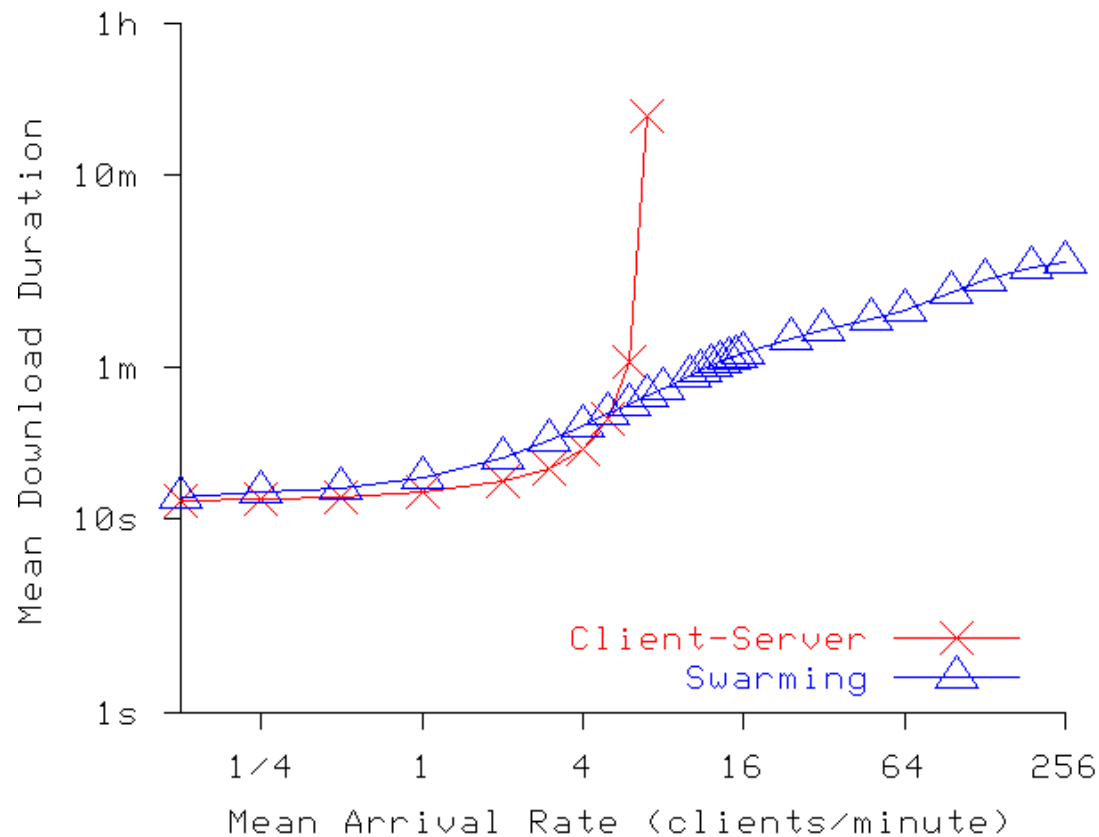
Simulation Methodology

- Packet-based, event-driven simulation
- Topology: Internet-as-a-router
 - Focus on bottleneck bandwidth
 - Varied client speeds
- Workload
 - For a fixed client arrival-rate – randomly generated client inter-arrival times using exponential distribution
 - Varied file sizes
- Metrics
 - Client download time, packet loss rate, etc
 - 5500 download completions per experiment



Scalability With Offered Load

- Several orders of magnitude beyond client-server
 - No lingering, server delivers at least 1 block to all clients
 - 192 clients/minute = 1 MB to ¼ million people/day



(Client-server would need
28 Mbps ~ \$5K-10K/month)

Details:

1 Mbps server

1536 Kbps/128 Kbps client

1 MB file

Parameters:

N_d (concurrent downloads): 4

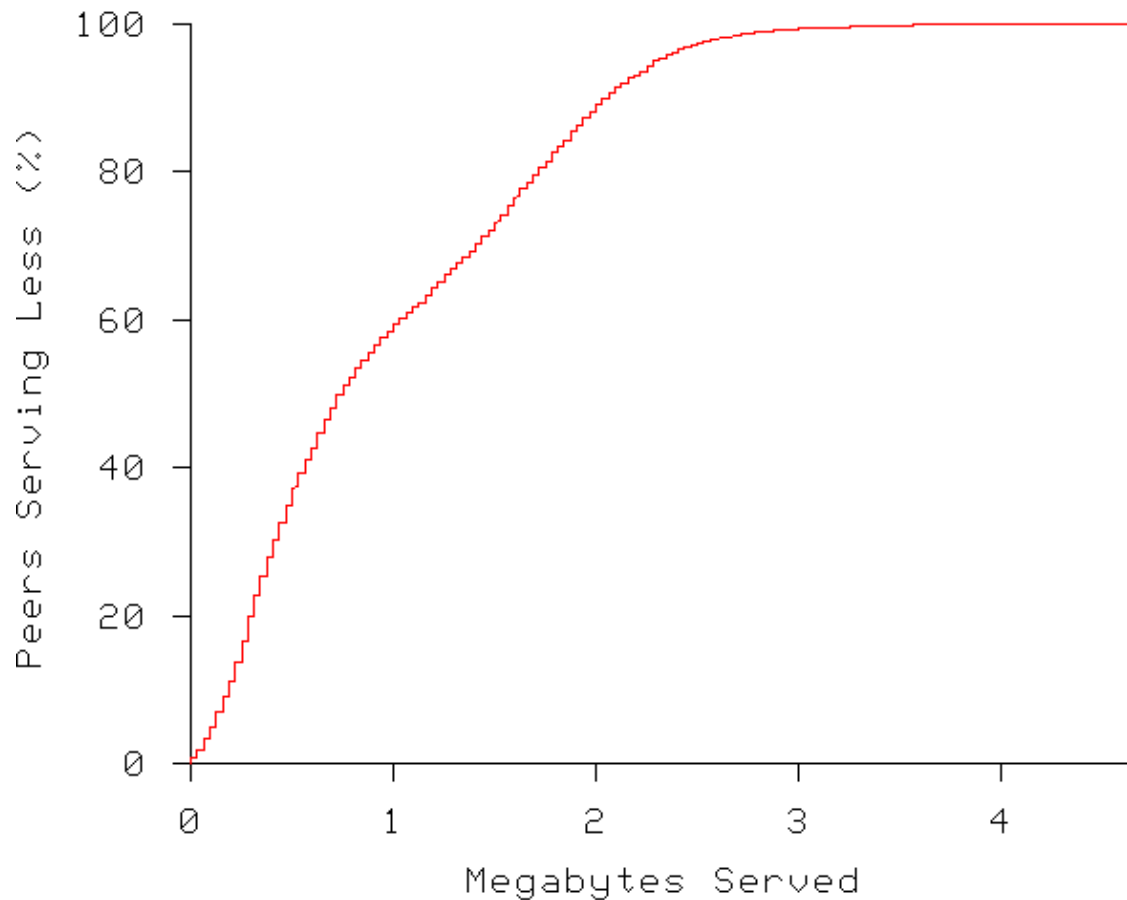
N_c (size of gossip cache): 64

N_g (peers in gossip message): 10

Block size: 32 KB

High Load: Load Balancing

- Roughly 60% of the clients serve less than 1 MB, nearly all of the clients upload less than 2 MB.



Details:

1 Mbps server

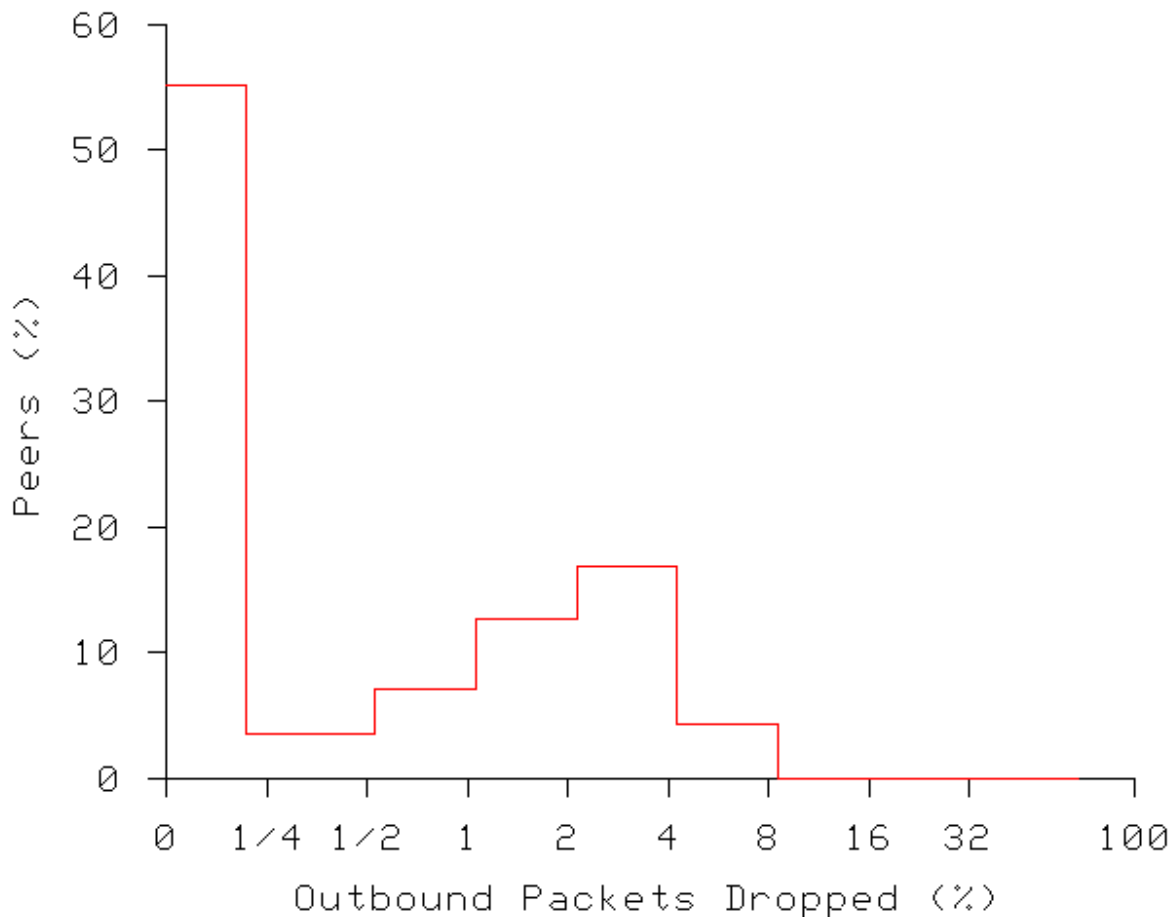
1536 Kbps/128 Kbps client

1 MB file

192 clients/minute

High Load: Minimal Packet Loss

- Server does have high loss but keeps swarming going
 - Client-server can't even operate in this region



Details:

1 Mbps server

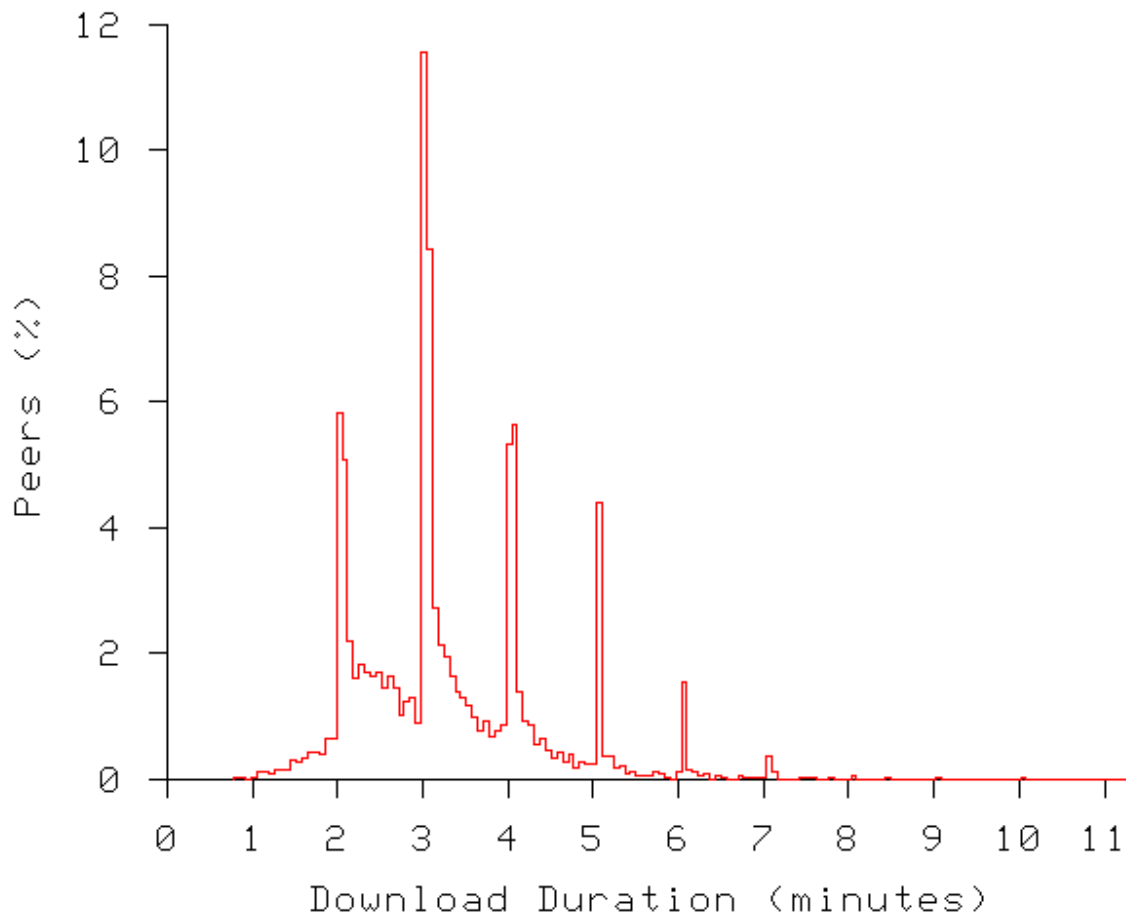
1536 Kbps/128 Kbps client

1 MB file

192 clients/minute

High Load: Download Times

- High variance in download time
 - Grouping at multiples of 60 caused by server congestion



Details:

1 Mbps server

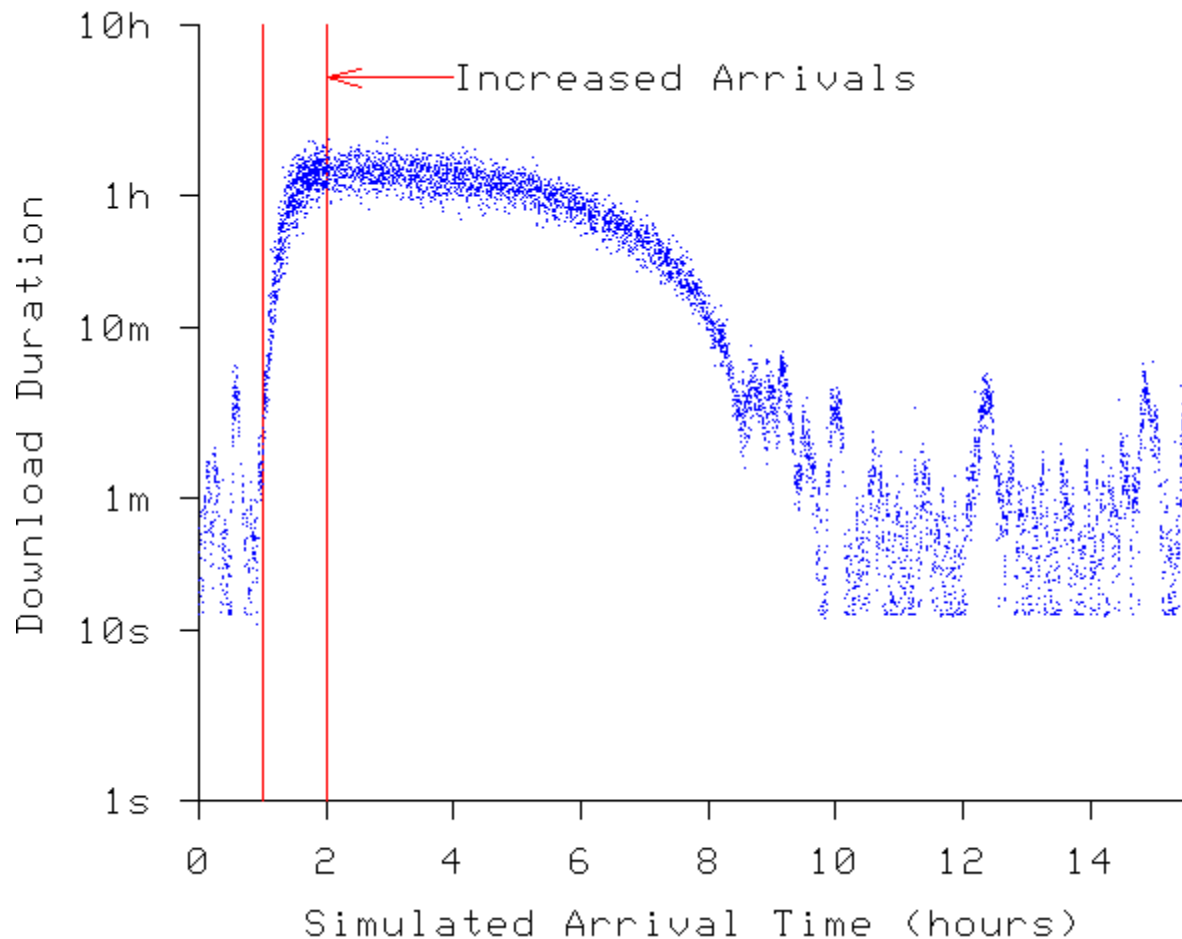
1536 Kbps/128 Kbps client

1 MB file

192 clients/minute

Flash Crowd – Standard Web Server

- Load swamps a standard web server



Details:

1 Mbps server
1536 Kbps/128 Kbps client
1 MB file

Client-Server Load:

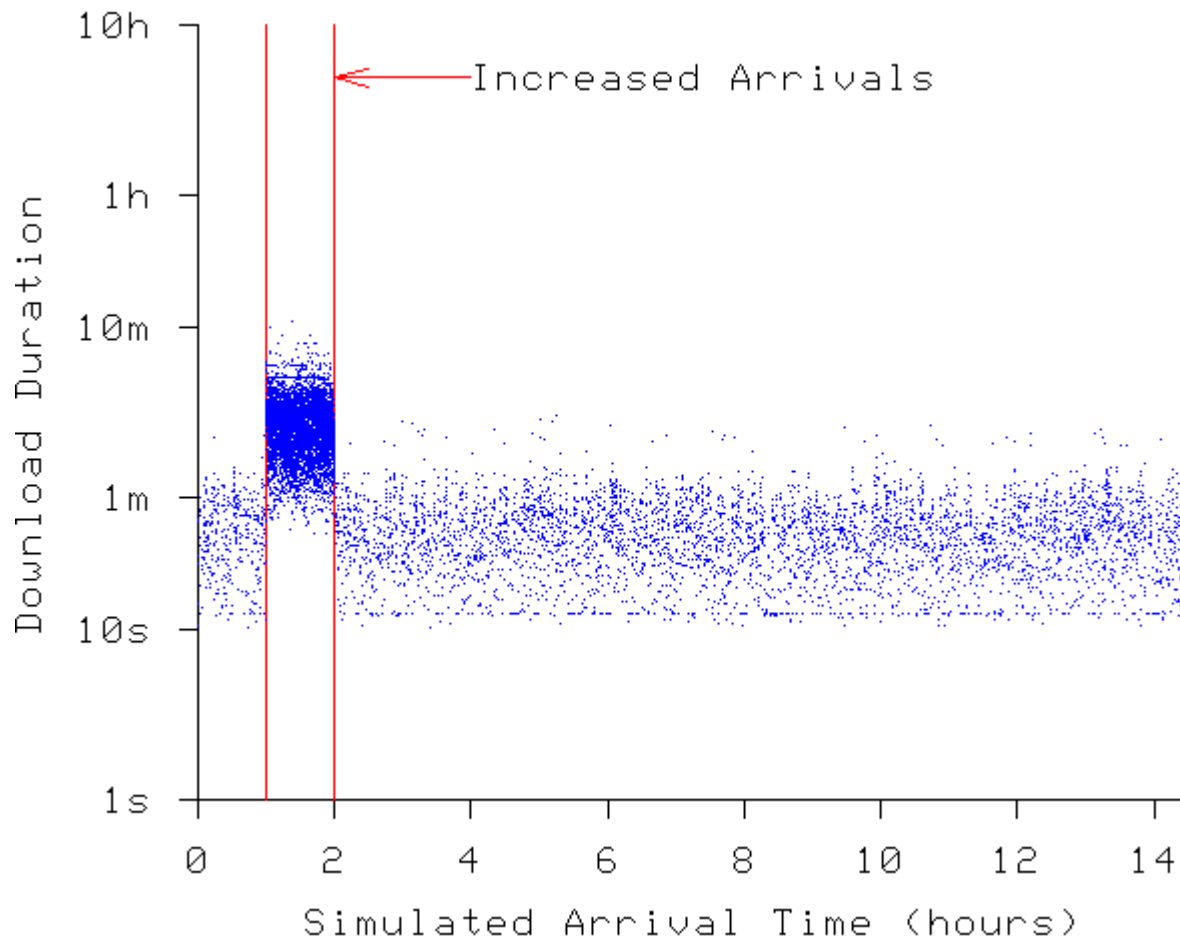
6 clients/minute for 1 hour
12 clients/minute for 1 hour
6 clients/minute

Swarming Load:

6 clients/minute for 1 hour
120 clients/minute for 1 hour
6 clients/minute

Flash Crowd - Swarming

- Minimal effect on swarming client performance



Details:

1 Mbps server
1536 Kbps/128 Kbps client
1 MB file

Client-Server Load:

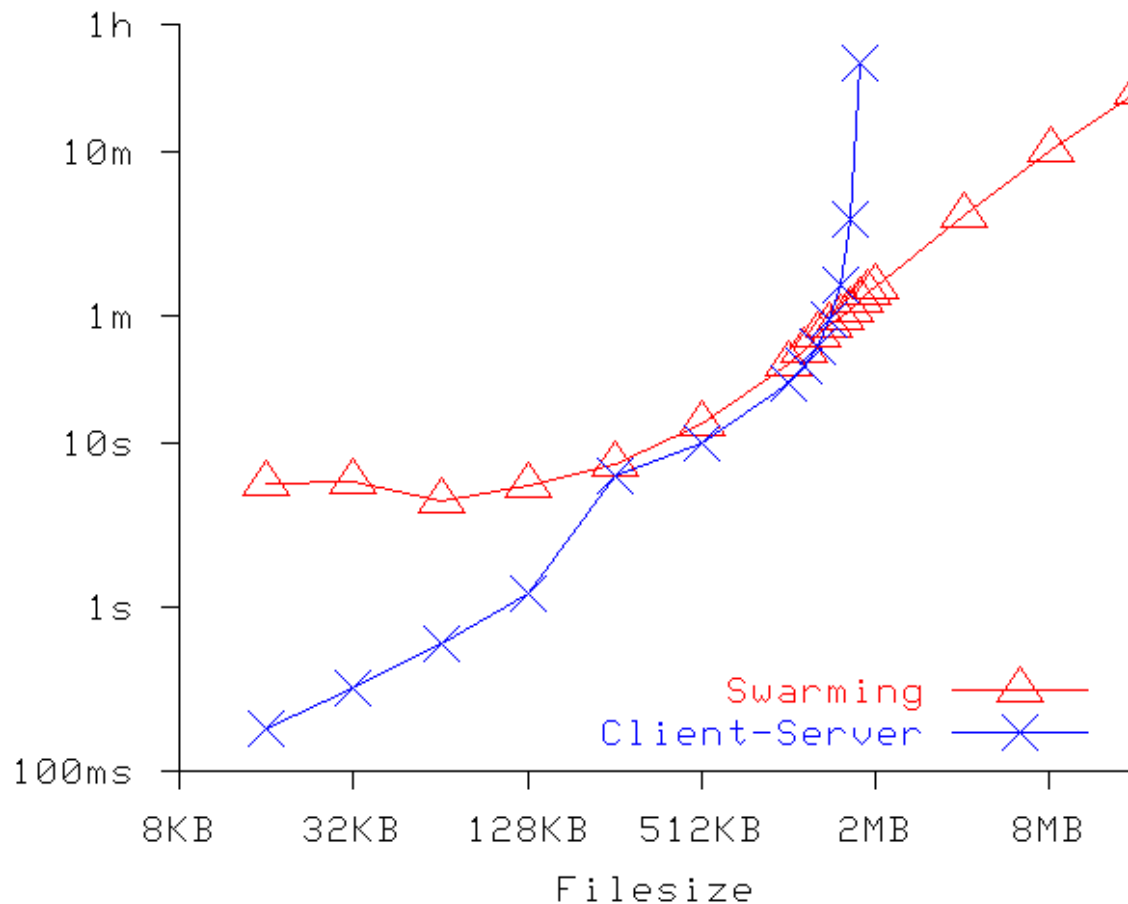
6 clients/minute for 1 hour
12 clients/minute for 1 hour
6 clients/minute

Swarming Load:

6 clients/minute for 1 hour
120 clients/minute for 1 hour
6 clients/minute

Scalability: File Size

- Linear scaling, small files have significant gossip overhead due to fixed number of blocks (32)



Details:

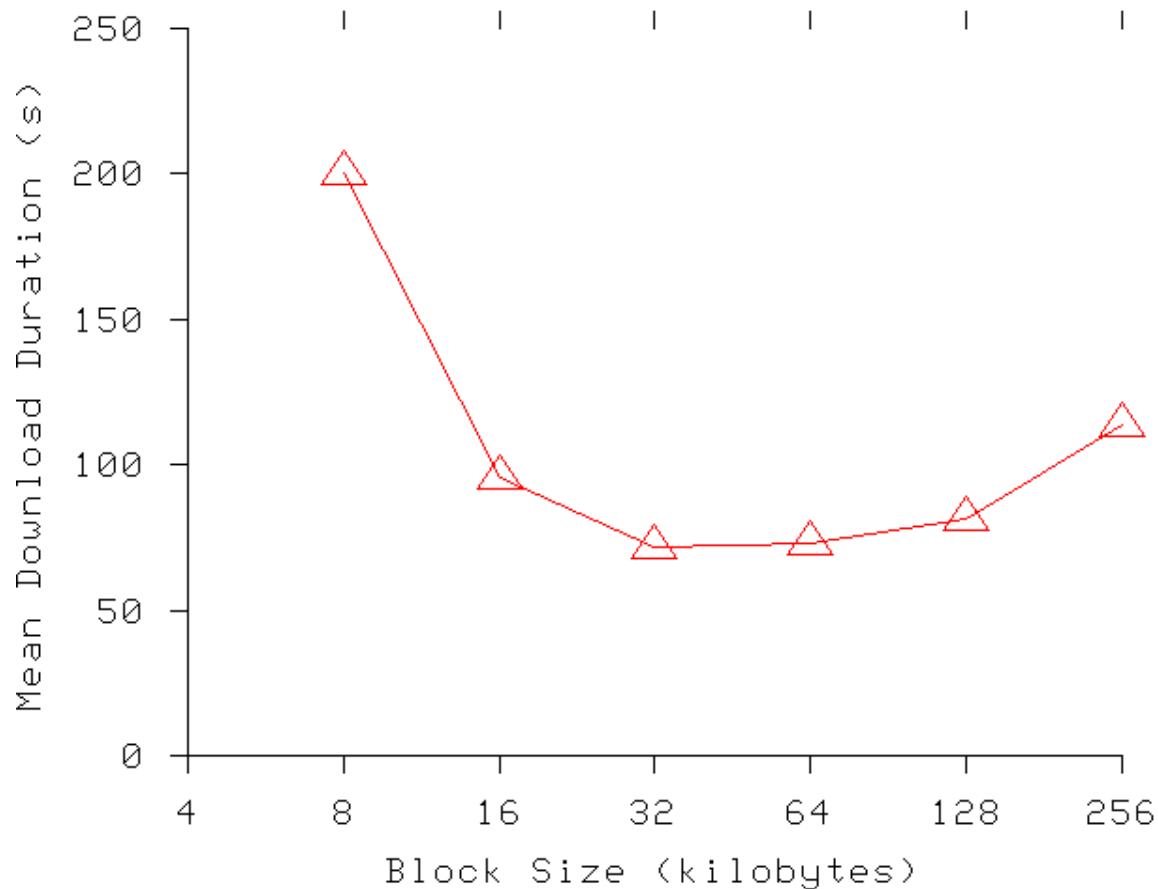
1 Mbps server

1536 Kbps/128 Kbps client

4 clients per minute

Scalability: Block Size

- Need large blocks – minimize gossip overhead
 - Too large – last block problem



Details:

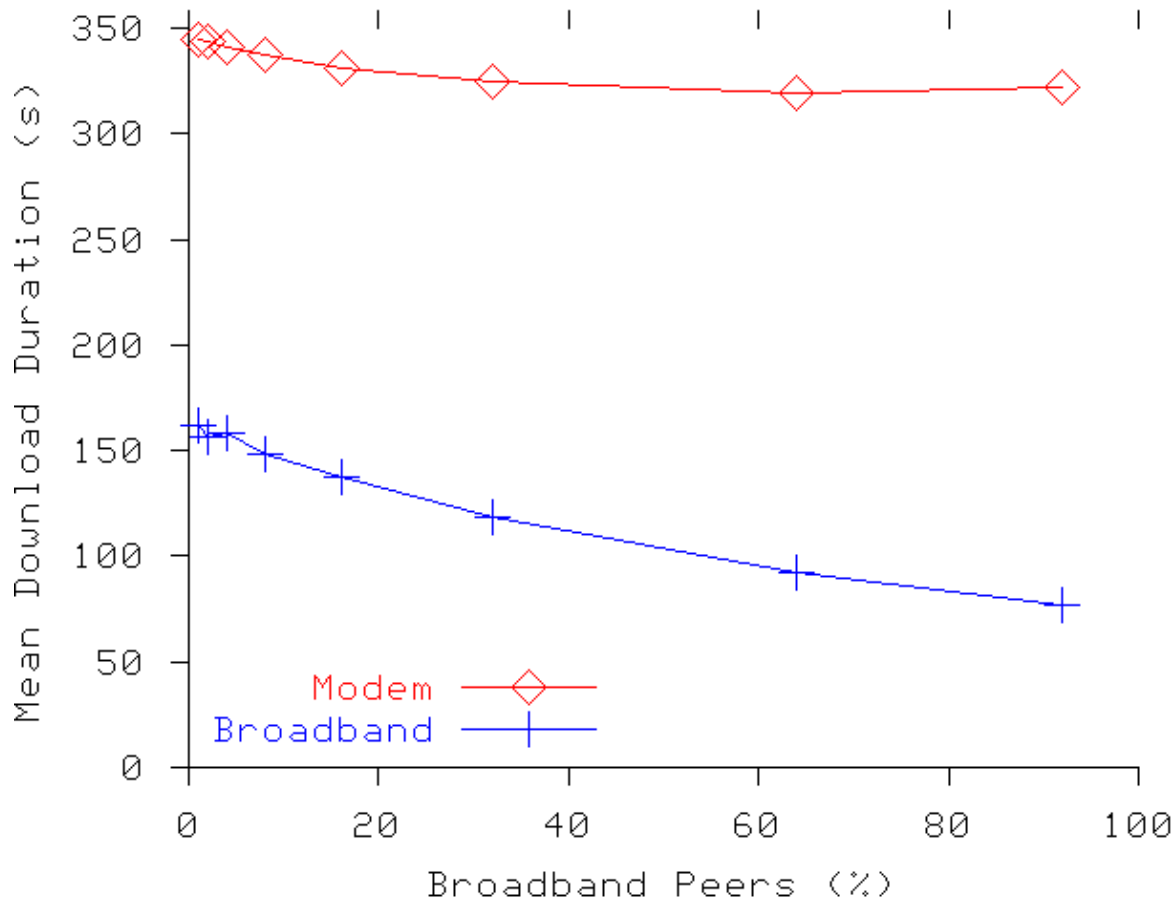
1 Mbps server

1536 Kbps/128 Kbps client

16 clients per minute

Scalability: Client Bandwidths

- Download time for broadband users doubles as modem users grow from 10% to 100%



Details:

1 Mbps server

1 MB file

16 clients/minute

Clients:

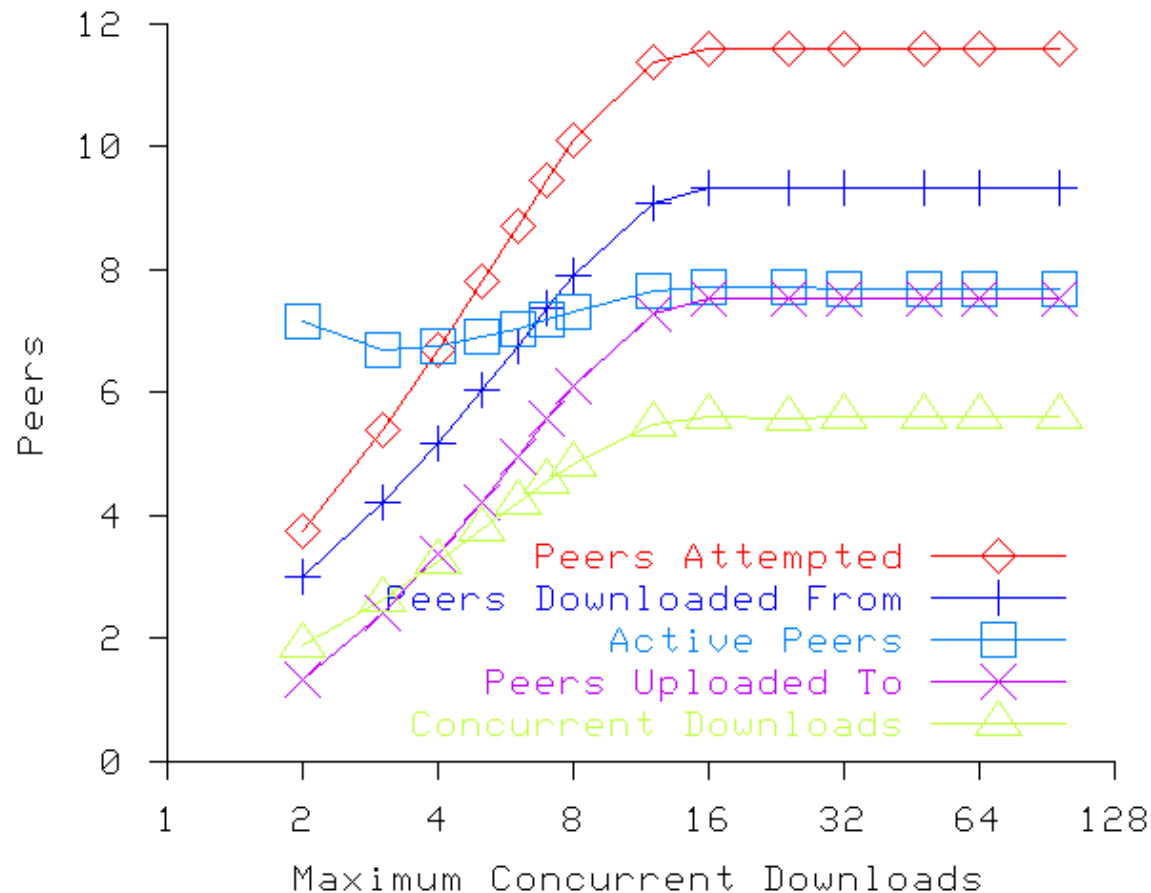
Office: 43 Mbps

Broadband: 1.5 Kbps/128Kbps

Modem: 56 Kbps/33 Kbps

Dynamics of Peer Selection

- But allowing more concurrent downloads spreads the load of file transfer among greater numbers of peers



Details:

1 Mbps server

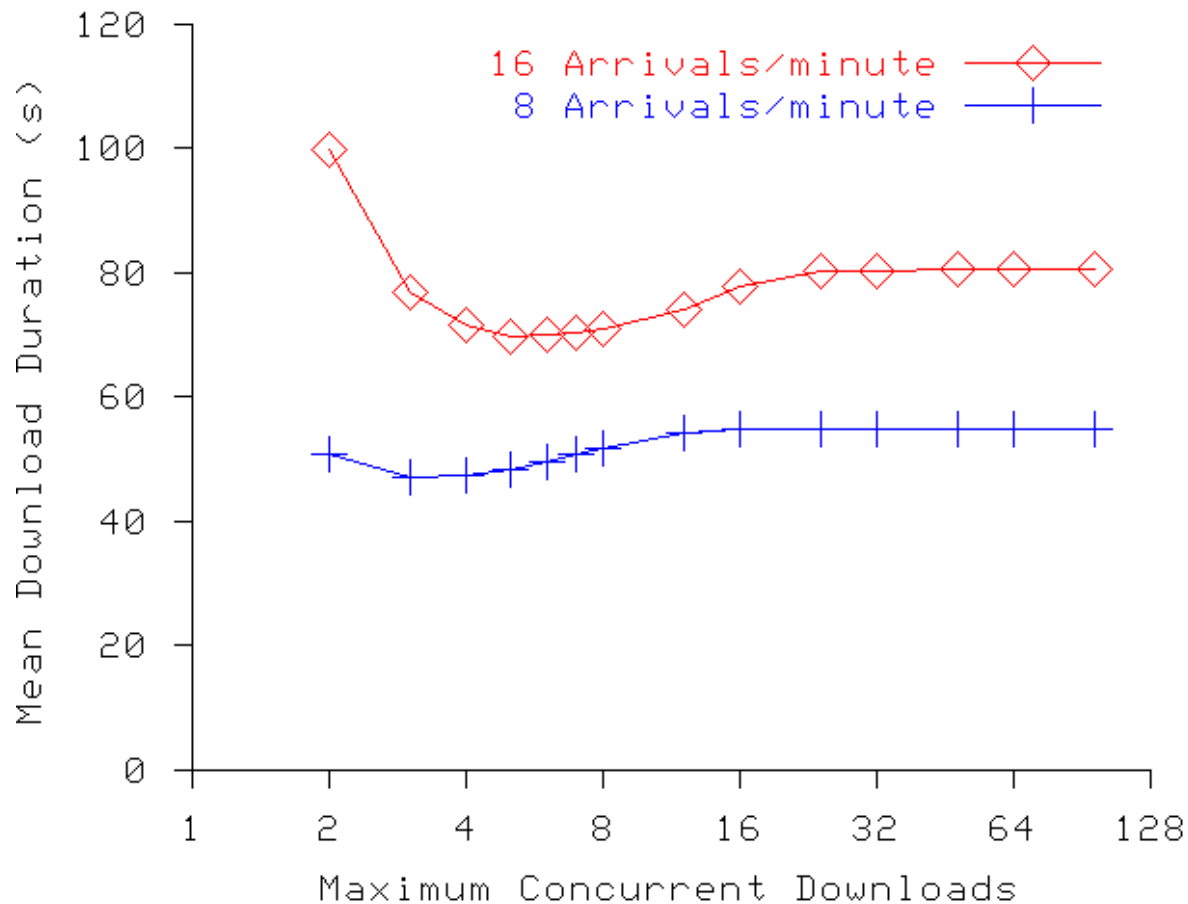
1536 Kbps/128 Kbps client

1 MB file

16 clients/minute

Concurrent Downloads

- Not a lot of concurrent downloads needed to speed up a download



Details:

1 Mbps server

1536 Kbps/128 Kbps client

1 MB file

Conclusions

- Swarming scales extremely well
 - load (arrival rate), file size, block size, client bandwidths
 - load is well-distributed, packet loss at clients is small
- Swarming easily handles flash crowds
- Impressive for a conservative, unoptimized implementation
- Need relatively large blocks (16 – 32 KB)
- Not taking full advantage of parallel download

Future Work

- Integrate swarming into a web server: dynamic swarming initiation to relieve server congestion at high load
- Encourage lingering to improve concurrency
- Better performance for small files
- Effects of non-cooperative peers and “super” peers
- Evaluation on a component basis – content location, peer selection, concurrent downloads, performance monitoring
- Head-to-head comparison with and detailed evaluation of BitTorrent both in simulations and on the PlanetLab testbed
- Examination of incentives in peer-to-peer content delivery