Slurpie: A Cooperative Bulk Data Transfer Protocol

J. L. Chiang

MNET Lab Meeting

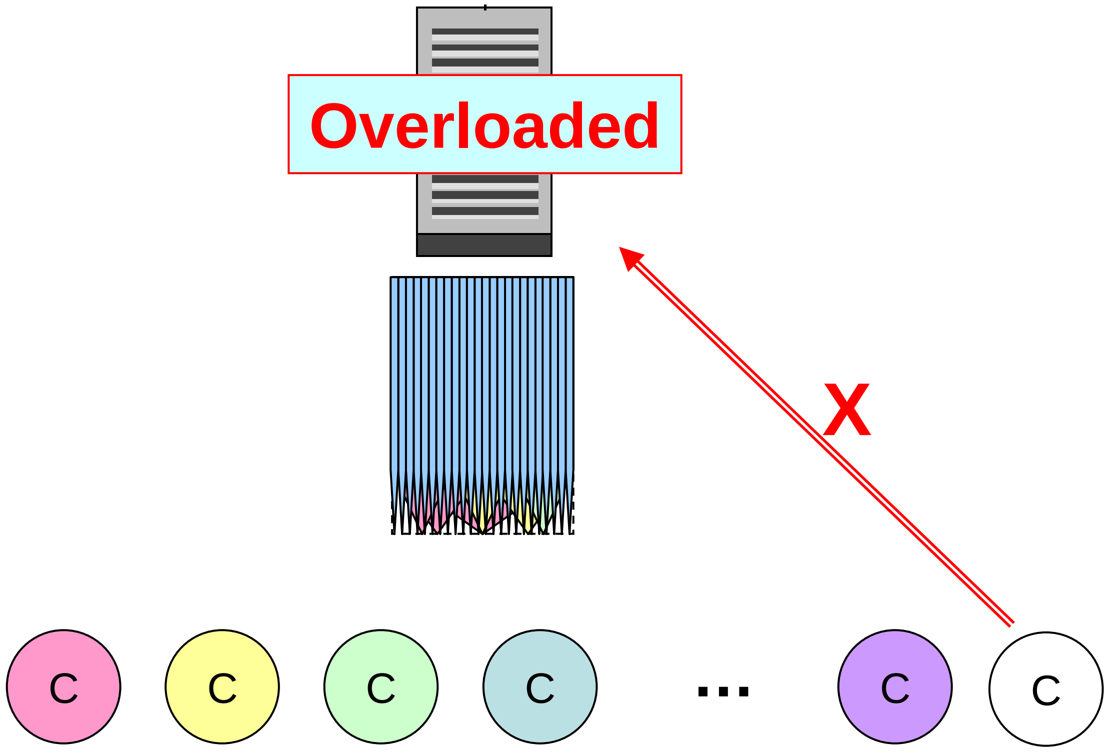
March 2, 2006

In Proc. INFOCOM, pp.941-951, 2004.

Outline

- Introduction
- Design goals and challenges
- The Slurpie Protocol
- Performance evaluation
- Conclusions and discussions

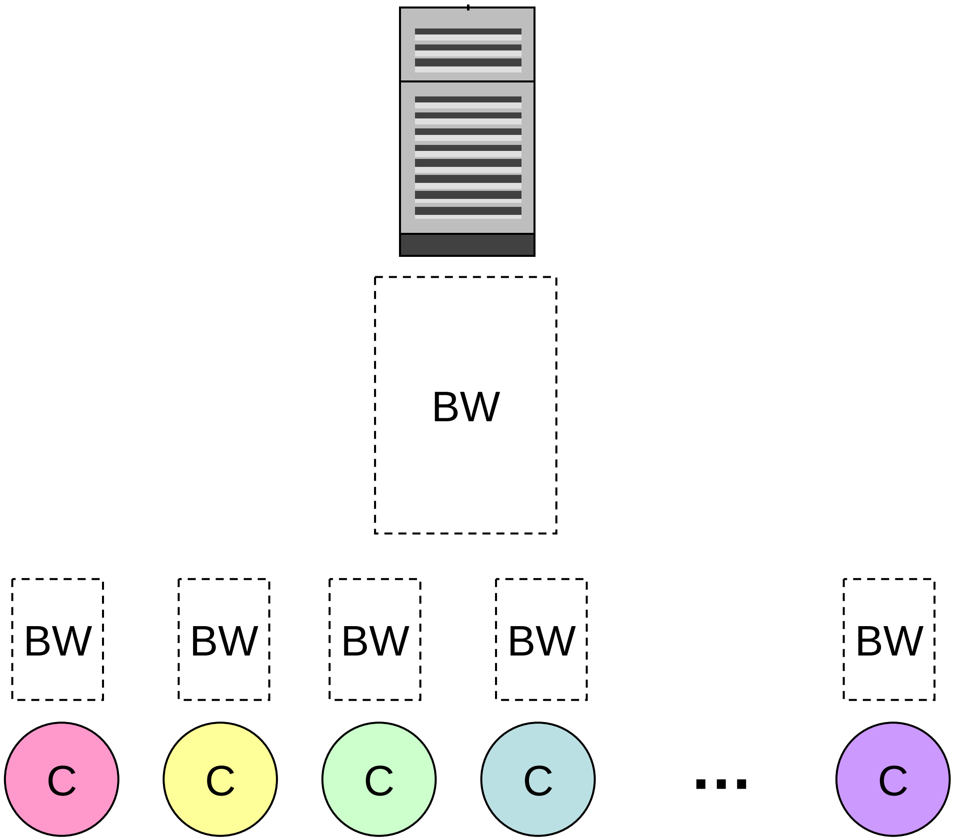
Introduction



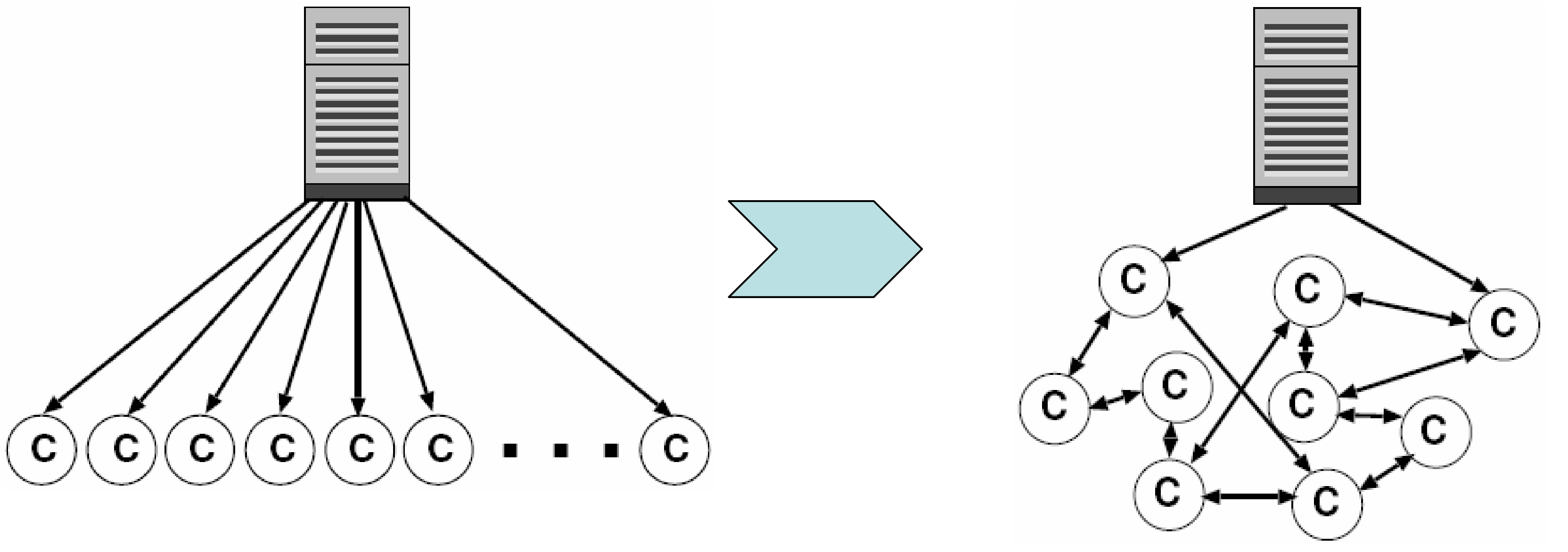
1 30mins 2 30mins 30mins

? mins/hours

Observations



Basic Idea



Design Goals

- Scalable
  - Maintain load at the server independent of the # of clients
- Beneficial
  - Minimized the client download times
- Deployable
  - Deployable w/o infrastructure support (except for a minimal loaded demultiplexing host)
- Adaptive
  - Adaptive download strategy based on network condition and client capacity
- Compatible
  - Require no server-side changes

Related Works

- Multicast
  - IP multicast, application layer multicast (EMS)

- **Focus on topology control**
- **Hard to adapt client dynamics**
- **Usually for long-term streaming**

- Infrastructure-based
  - CDN, web caches, mirrors

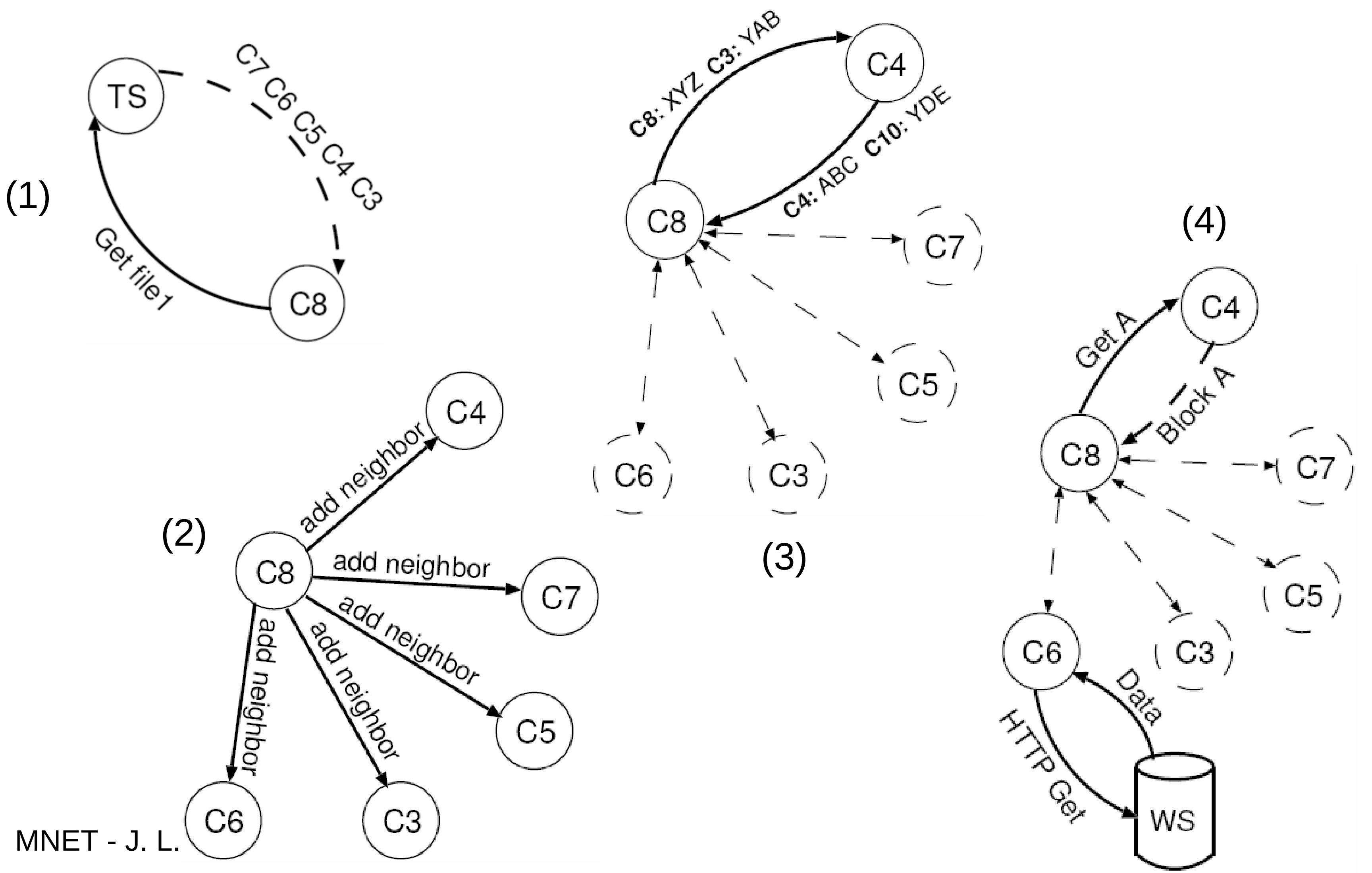
- **Explicitly provisioned for only certain load levels**

- Peer-to-peer
  - CoopNet, BitTorrent

- **Design for small HTML files**
- **All transfer involve the server**

- **Not adapt to varying bandwidth conditions**
- **Not scale its # of neighbors as the group size increases**
- **Tracker limits the scalability to thousands of nodes**

The Slurpie Protocol



Challenges

- Maintain exact state about all peers downloading a file
- How the mesh is formed
- How updates are propagated
- Decide whether to contact server or peers
- How many blocks to divide the files into
- How many connections each peer should open

Mesh Formation

- TS maintains and returns the last φ seed nodes that queried the TS for that same file.
  - φ is a small constant.
- The newly joined node makes bidirectional links to a random set of these seed nodes, and tries to discover η neighbors to maintain
  - η is updated depending on available bandwidth.
  - $\eta \geq O(\log n)$ to ensure that the mesh stays connected with high probability.
  - n is the estimated number of nodes in the mesh.

Group Size Estimation

- From random graph theory, for an *r*-regular graph, the mean distance *d* between nodes is proportional to $log_{r-1}n$.
- n = $O((r-1)^d)$
  - Use *U* updates to estimate *d* by *hop_count* and *r* by *degree*.

n	20	50	100
200	17.5%	13.2%	10.9%
1000	5.9%	4.2 %	2.6%
5000	11.3%	7.5%	6.0%
10000	3.8%	0.8 %	0.4%

TABLE II

% ERROR IN GROUP SIZE ESTIMATION

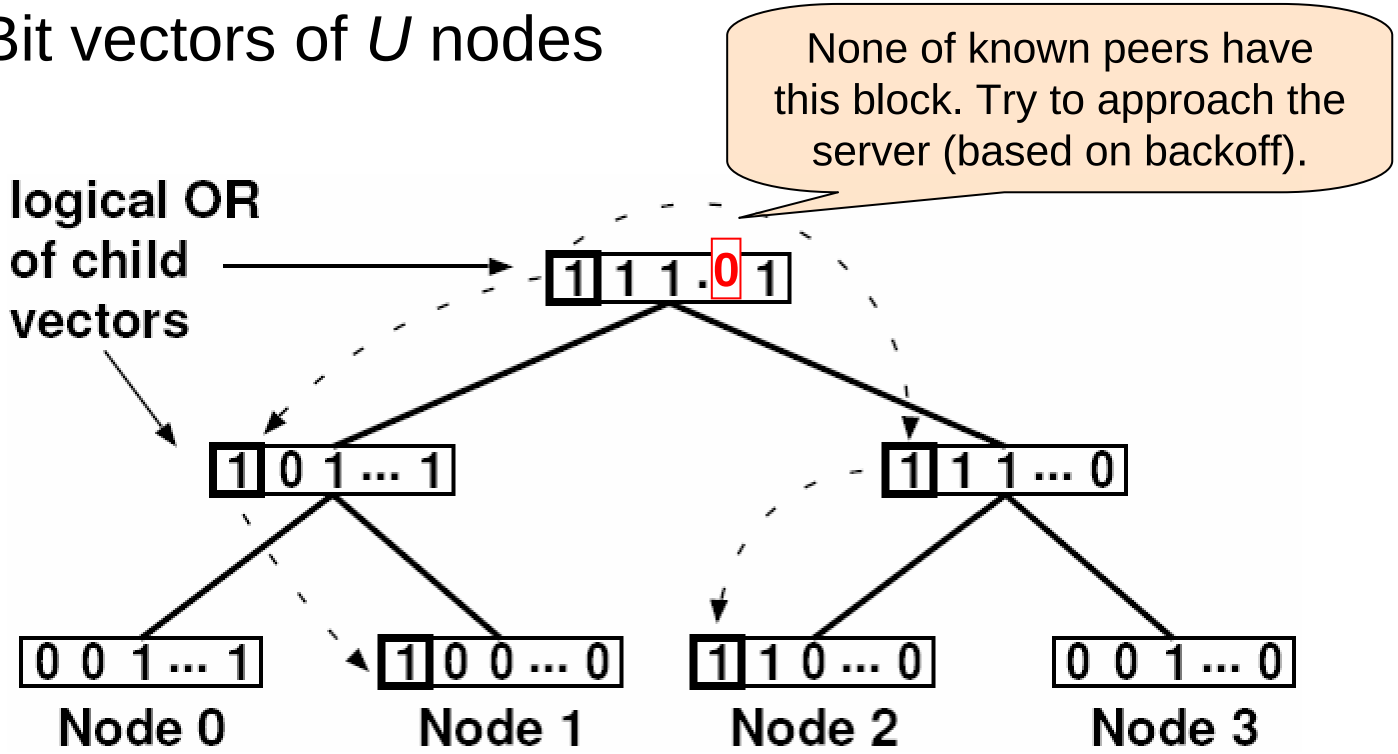
Update Propagation (1/2)

- The update message:
  - <IP-addr, port, block-list, hopcount, node-degree>

- The rate of updates passed along each link per second, σ , is subject to an AIMD flow control algorithm based on available bandwidth estimates.

Update Propagation (2/2)

- The update tree:
  - Bit vectors of *U* nodes



Bandwidth Estimation (1/2)

- Three states:
  - *underutilized*, *at-capacity*, *throttle-back*

- *Bmax*: coarse grained bandwidth setups
  - “Modem”, “T1/DSL”, “T3”, …
- *Bact*: actual achieved throughput over all data connections over a 1s interval
- *avgB*: moving average of *Bact*, along with a *std*

Bandwidth Estimation (2/2)

- *Underutilized*:
  - *Bact* < *Bmax* – *std*
  - η ++, σ ++, data_connection ++
- *Throttle-back*:
  - *Bact* > *avgB* + *std*
  - η --, σ /2, data_connection --
- *At-capacity*:
  - Others
  - No change on system parameters

Download Decisions

- When multiple peers have the same block, a peer is chosen at random.
- A node prefers to download from the peer with established connection to take advantage of an open TCP window.
- The bandwidth estimation algorithm is queried every second, if it returns *underutilized*, and there exist hosts that have blocks that the local node does not have, a new connection is opened.

Random Backoff (1/2)

- Control the load on the source server independent of peers in the system.
- Ideally, the host with the best connection to the server would be the sole machine connected to the server, and everyone else receive their data from this host.
  - Finding the best host is difficult.
  - The best host could download the data and then leave.

Random Backoff (2/2)

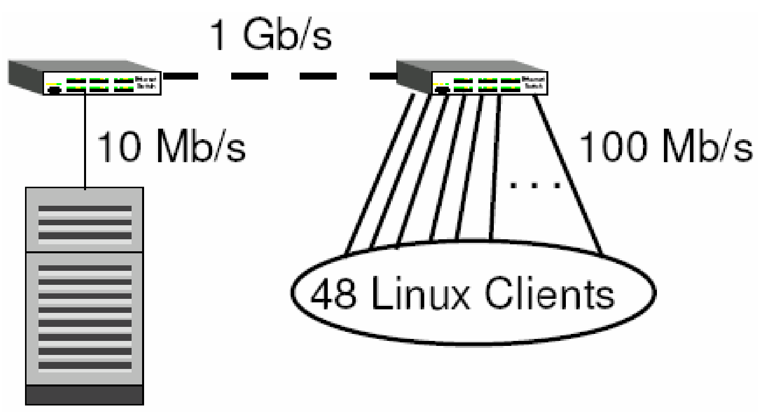
- Every time period τ , each eligible peer decides to go to the server with probability *k*/*n*.
  - *n* is the estimate of the nodes in the system.
  - *k* is a small constant.
  - (setting *k*=1 results in no connections at the server for about 30% of extended periods of time)
  - (Setting *k*=3 keeps at least one connection at the server about 90% of the time)
  - τ is chosen to be long enough for progress but short enough for fairness.

Block Size

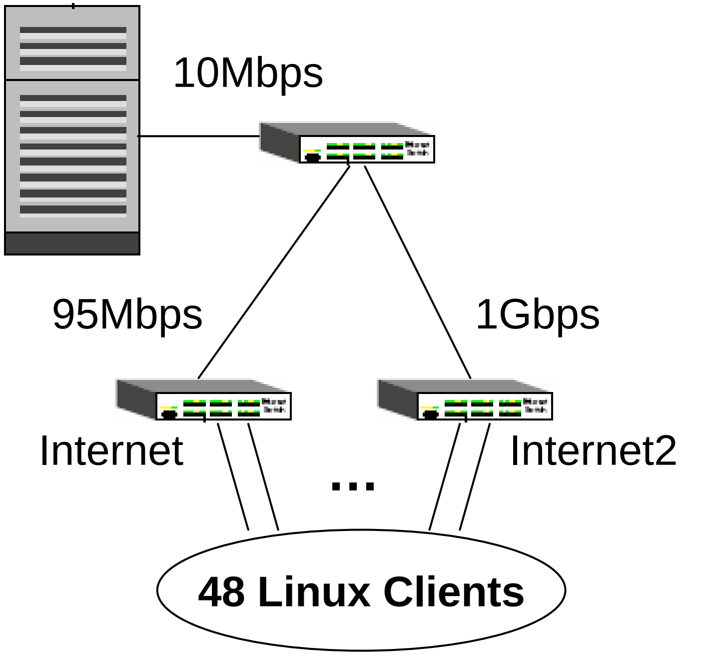
- Fixed 256KB block size is chosen.
  - It is the smallest size at which the TCP overhead was effectively amortized (<1%).
  - It keeps the bit vector to a manageable size for large files (50 bytes for a 100MB file).

Experimental Setup

- Local Testbed Setup



- PlanetLab Setup



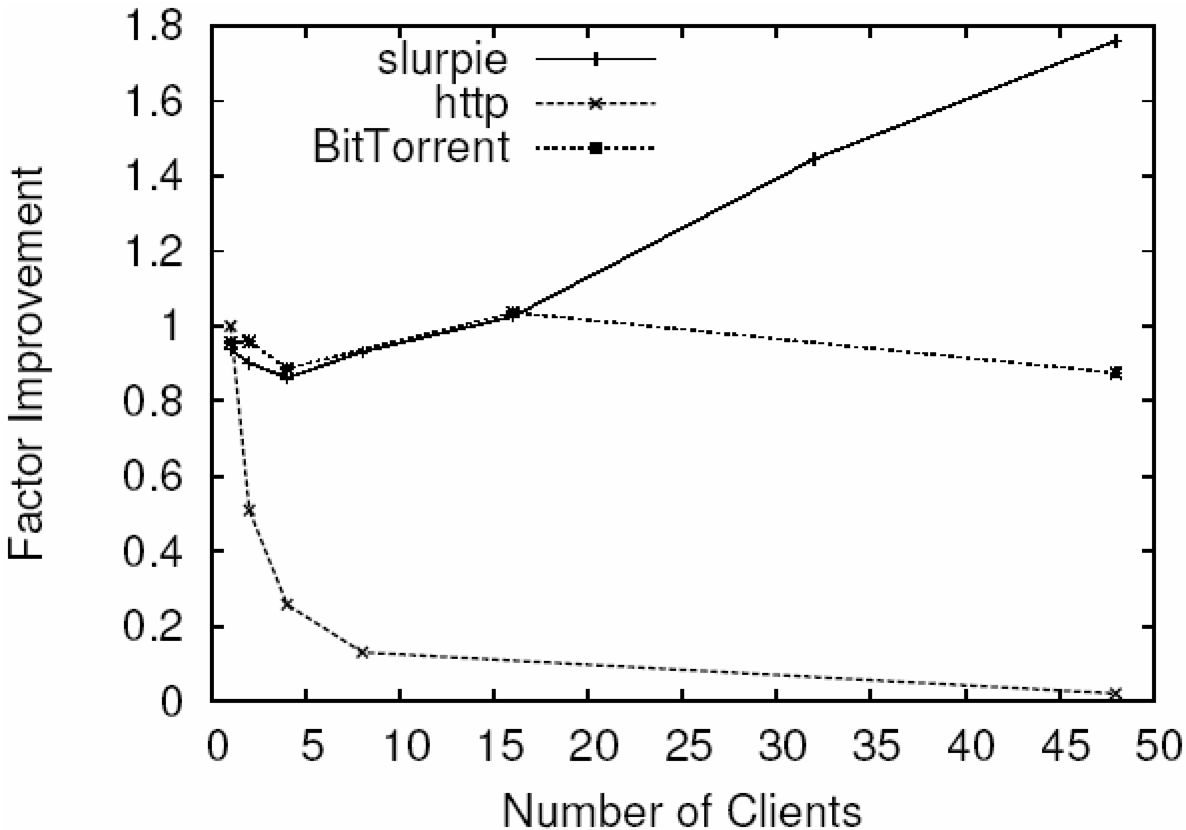
Parameter Setup

Parameter	Description	Value
k	k/n clients go to server	3
τ	Server connection length	4 seconds
σ	Initial Update Rate	8/second
η	Initial Number of Neighbors	10
m	Mirror Time (described below)	2 seconds
U	Number of Updates Stored	100
ψ	Per File State at Topology Server	5

TABLE I

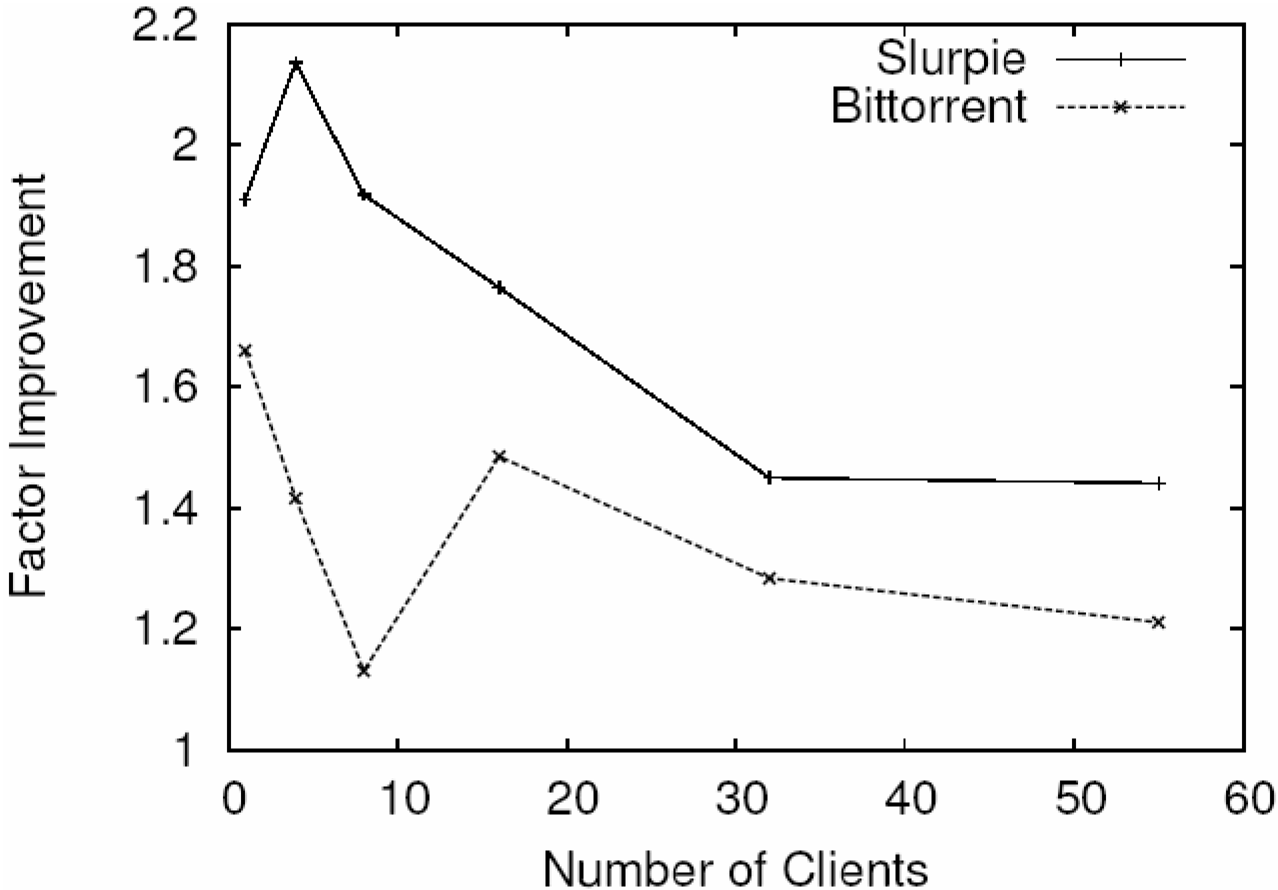
DEFAULT SLURPIE PARAMETERS

Completion Times (Local)

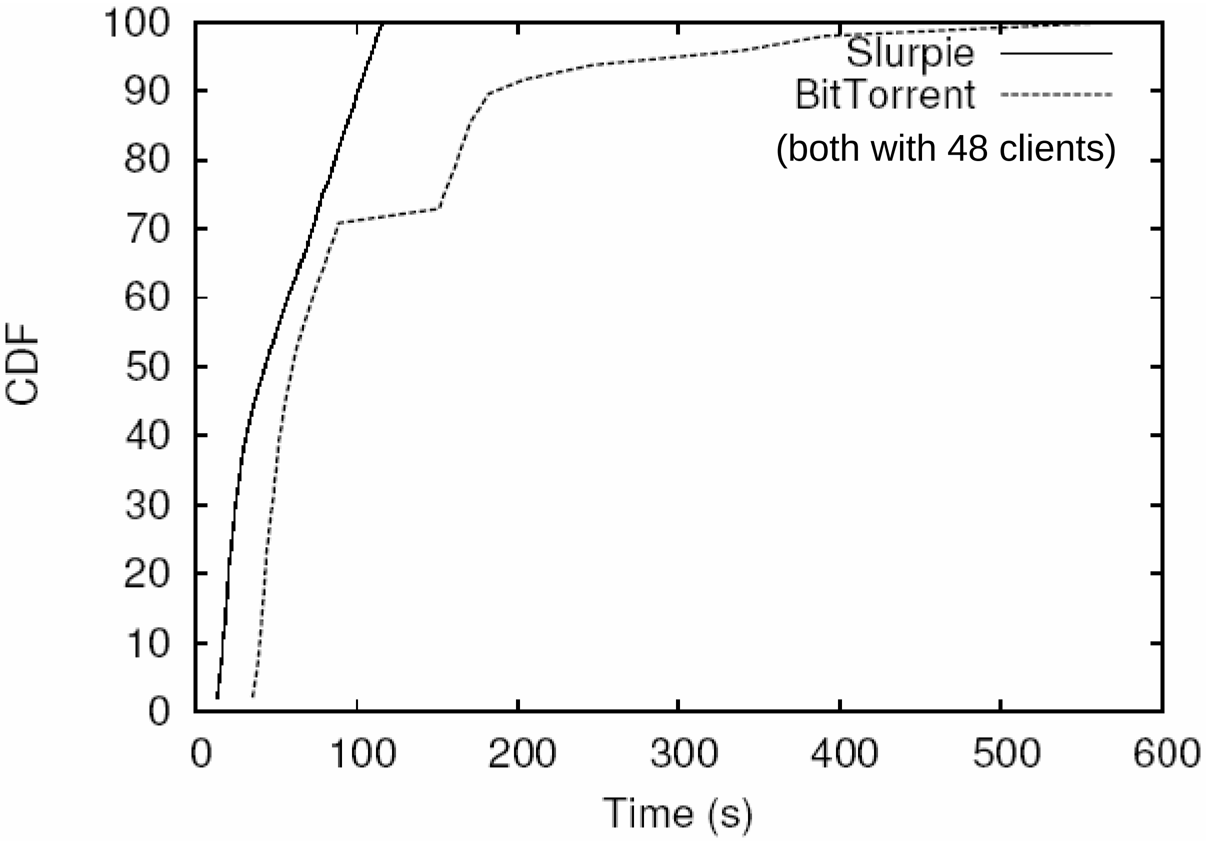


Factor = experiment time / baseline time

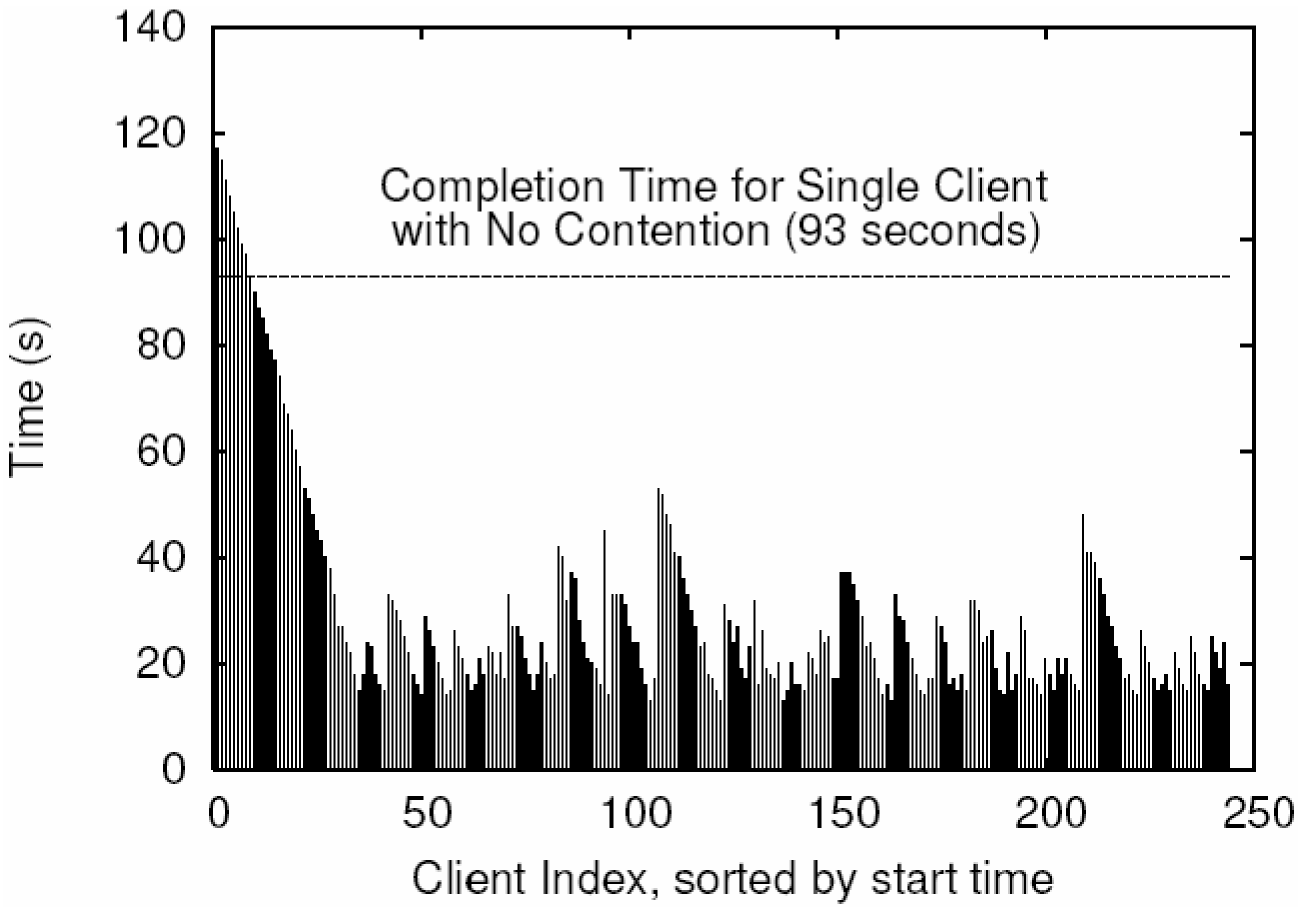
Completion Times (PlanetLab)



CDF of Completion Times

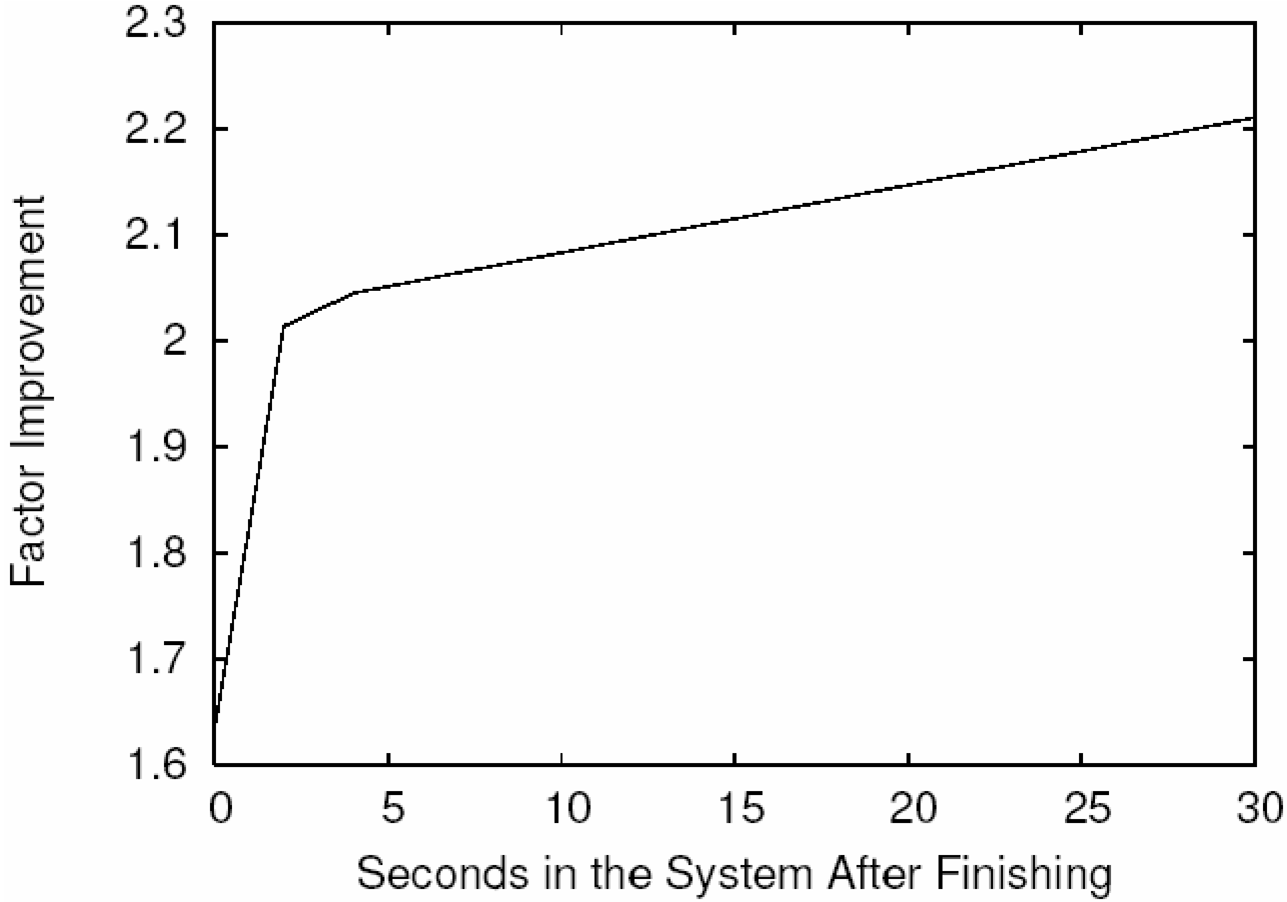


Completion time of Client

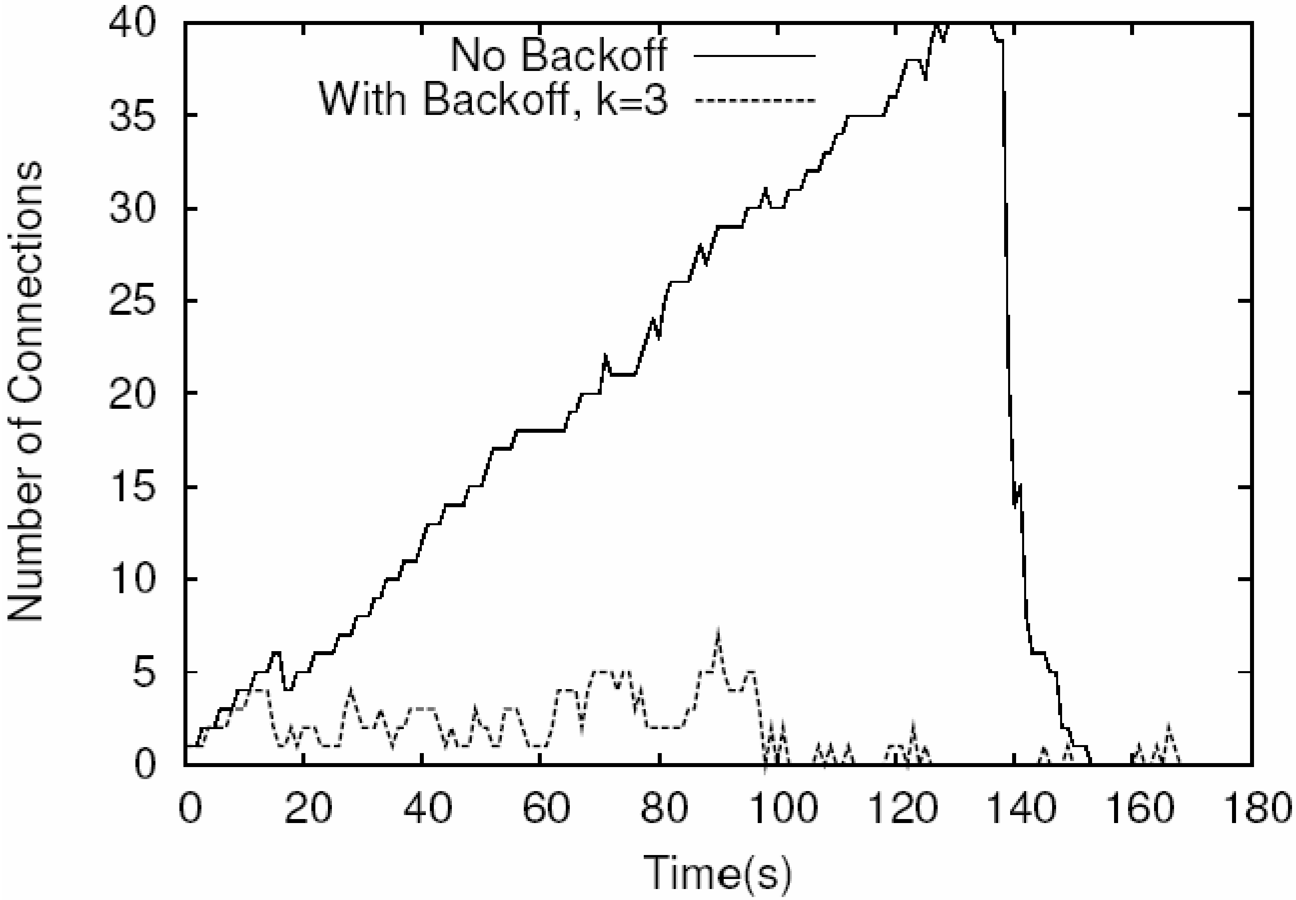


Slurpie w/ 245 client, each arrives 3s after previous one

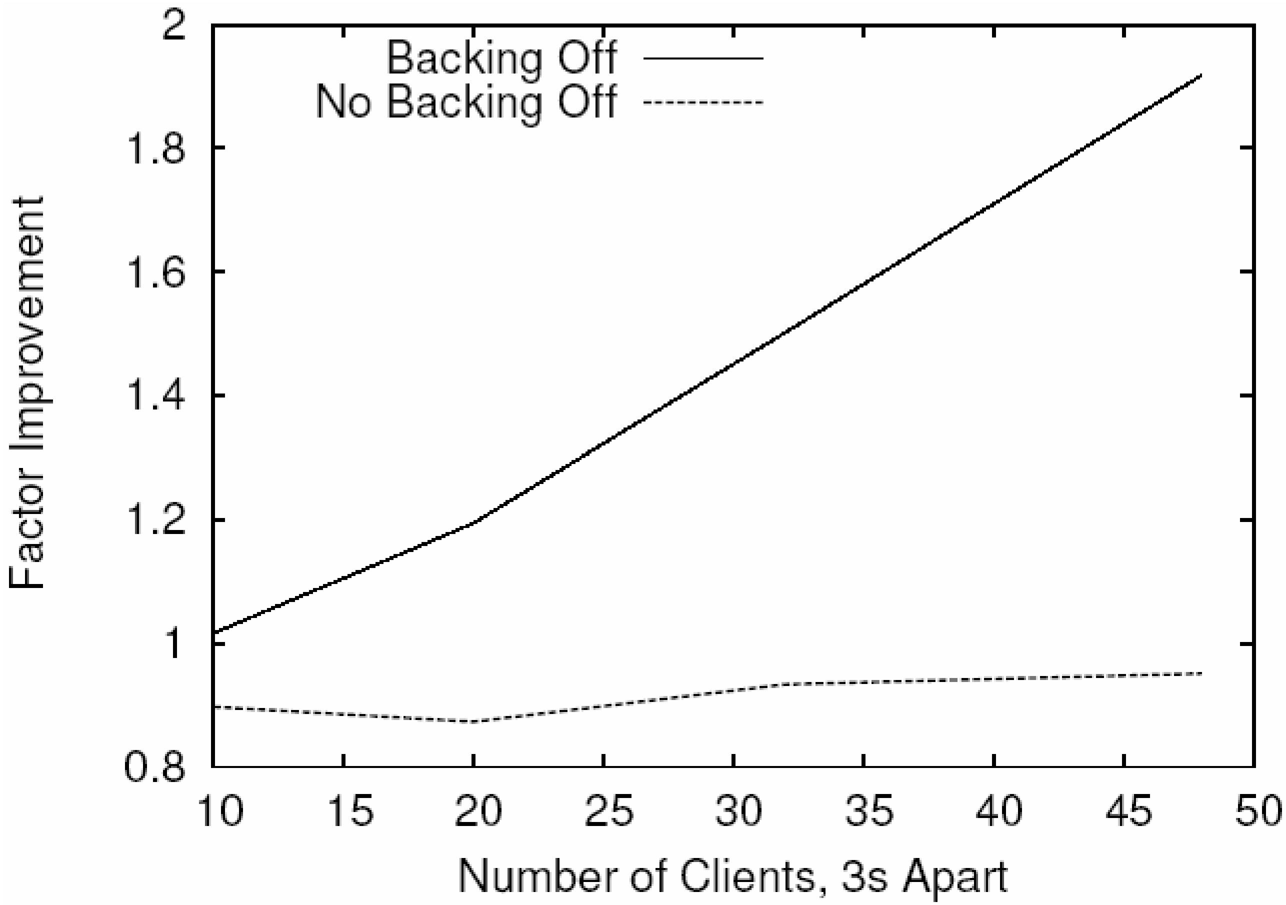
Effect of Benevolence



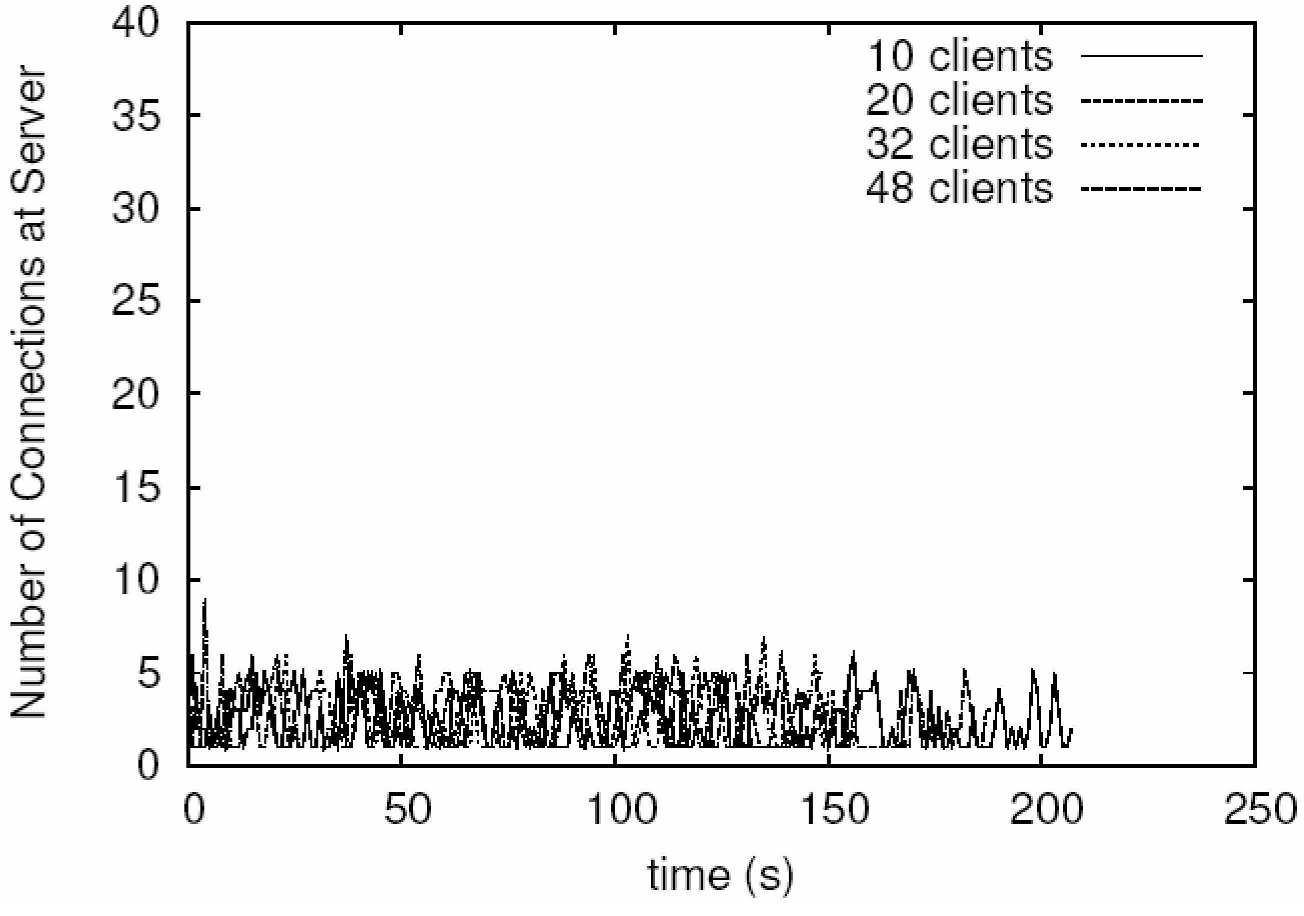
Effect of Backoff on Server



Effect of Backoff on Client



Effect of Flash Crowds

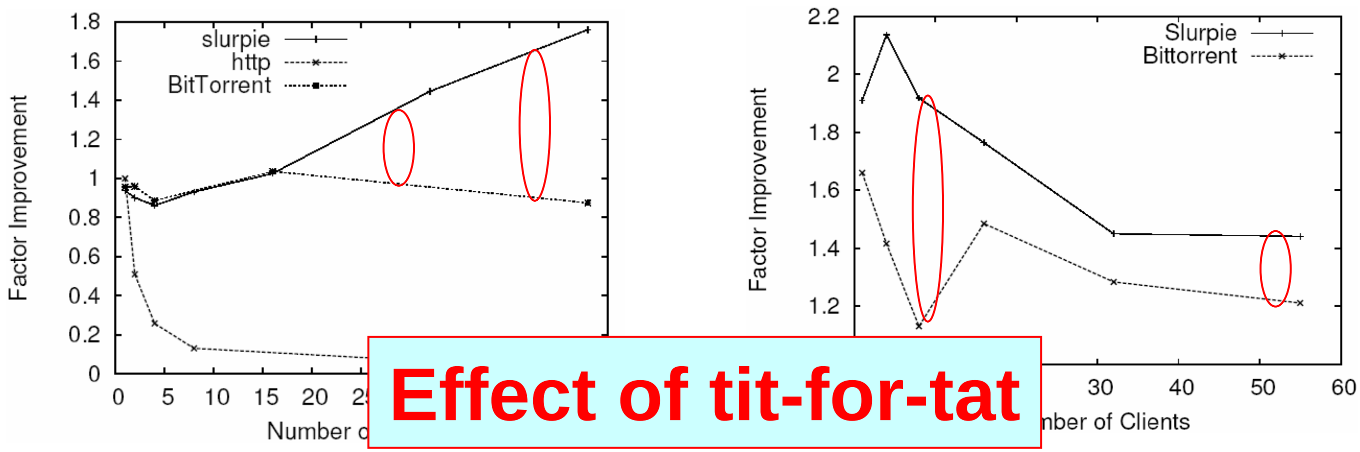


Summary

- Slurpie protocol fulfills the design goals of system scalability, improved client performance and insulation of the server from load variance in the client population.
- Future works
  - Internet-wide deployment
    - initial mirror sites, slurpie proxy
  - Better estimation of network size

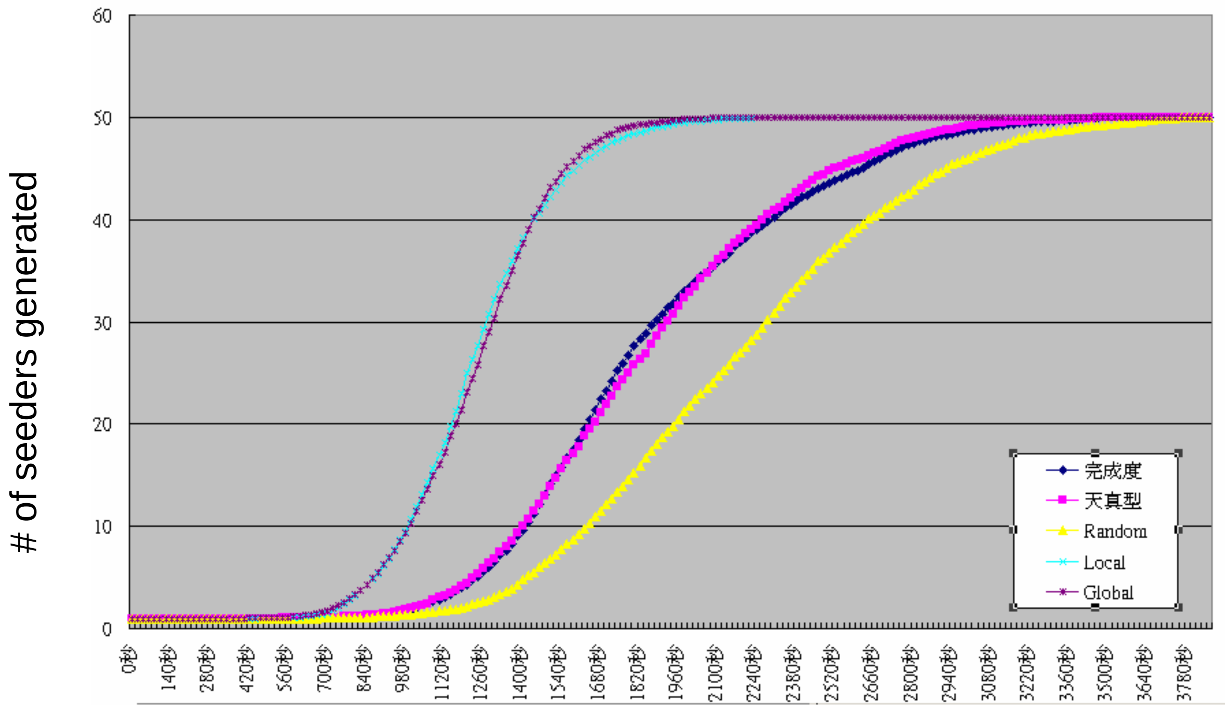
Discussions (1/3)

- Clients download parts of files from other clients without accessing highly contested server resources.
- BT v.s. Slurpie



Discussions (2/3)

- Effect of piece (block) selection algorithms
  - Last block problem, peer utility



Discussions (3/3)

- A larger # of clients should be experimented.
- The overhead of update propagation should be considered.
- Which data connection should be closed when *throttle-back*.