Slurpie

A Cooperative Bulk Data Transfer Protocol

Rob Sherwood   Ryan Braud   Bobby Bhattacharjee

University of Maryland

Problem

- Motivation

Problem

- Motivation
  - High bandwidth client and server

Problem

- Motivation
  - High bandwidth client and server
  - Internet core bandwidth under-utilized

Problem

- Motivation
  - High bandwidth client and server
  - Internet core bandwidth under-utilized
  - ... but downloading popular files is still slow

Problem

- Motivation
  - High bandwidth client and server
  - Internet core bandwidth under-utilized
  - ... but downloading popular files is still slow
- Mitigating Factors
  - Usage patterns difficult to predict
    - slashdot effect, popularity spikes, etc..
  - Competing TCP Streams result in suboptimal performance

Goals

- Decrease download times for large, popular files

Goals

- Decrease download times for large, popular files
- Reduce load at the server

Goals

- Decrease download times for large, popular files
- Reduce load at the server
- Should not require server-side modification

Goals

- Decrease download times for large, popular files
- Reduce load at the server
- Should not require server-side modification
- Compatible with existing protocols;
  e.g. http/ftp/etc..

Goals

- Decrease download times for large, popular files
- Reduce load at the server
- Should not require server-side modification
- Compatible with existing protocols;
  e.g. http/ftp/etc..
- Scalable into 10^4-10^6 nodes

Assumptions

- Source server is the bottleneck

Assumptions

- Source server is the bottleneck
- A small number of popular files represents a disproportionate amount of traffic

Assumptions

- Source server is the bottleneck
- A small number of popular files represents a disproportionate amount of traffic
- Peers are able and willing to share content

Assumptions

- Source server is the bottleneck
- A small number of popular files represents a disproportionate amount of traffic
- Peers are able and willing to share content
  - If it is to their benefit

Assumptions

- Source server is the bottleneck
- A small number of popular files represents a disproportionate amount of traffic
- Peers are able and willing to share content
  - If it is to their benefit
  - If the cost is negligible

Assumptions

- Source server is the bottleneck
- A small number of popular files represents a disproportionate amount of traffic
- Peers are able and willing to share content
  - If it is to their benefit
  - If the cost is negligible
  - Peers do not want to persist in the network indefinitely

Assumptions

- Source server is the bottleneck
- A small number of popular files represents a disproportionate amount of traffic
- Peers are able and willing to share content
  - If it is to their benefit
  - If the cost is negligible
  - Peers do not want to persist in the network indefinitely
- End-to-End data integrity check available, e.g. md5sum

Partial Solution

Partial Solution

- Peers download small, random subsections, *chunks*, for the source server

Partial Solution

- Peers download small, random subsections, *chunks*, for the source server
- All peers downloading a specific file form a random mesh

Partial Solution

- Peers download small, random subsections, *chunks*, for the source server
- All peers downloading a specific file form a random mesh
- Peers propagate which chunks they have, e.g. *updates*, through the mesh

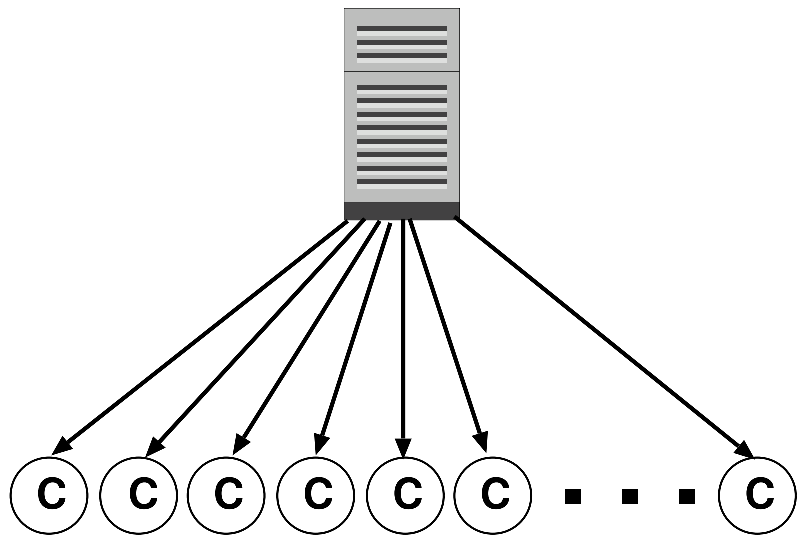
Partial Solution

- Peers download small, random subsections, *chunks*, for the source server
- All peers downloading a specific file form a random mesh
- Peers propagate which chunks they have, e.g. *updates*, through the mesh
- Peers exchange chunks, i.e. p2p

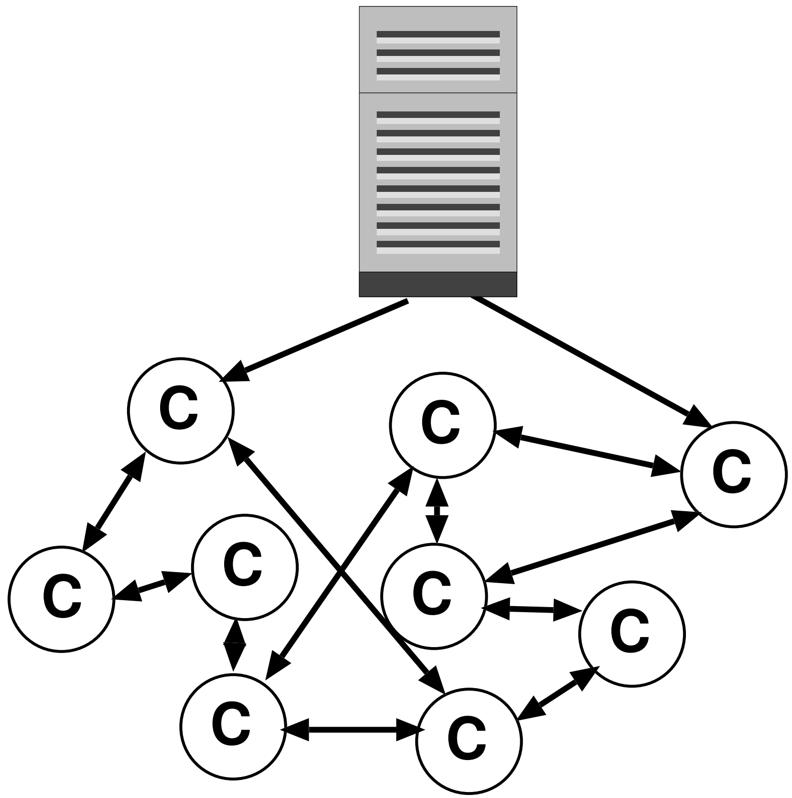
Partial Solution

- Peers download small, random subsections, *chunks*, for the source server
- All peers downloading a specific file form a random mesh
- Peers propagate which chunks they have, e.g. *updates*, through the mesh
- Peers exchange chunks, i.e. p2p
- Peers leave the mesh/system as soon as they complete the file

Partial Solution



Partial Solution



Full Solution: Outline

Full Solution: Outline

- Use *topology server* to coordinate peers

Full Solution: Outline

- Use *topology server* to coordinate peers
- Random graph model for mesh

Full Solution: Outline

- Use *topology server* to coordinate peers
- Random graph model for mesh
- Bandwidth estimation to optimize update propagation overhead

Full Solution: Outline

- Use *topology server* to coordinate peers
- Random graph model for mesh
- Bandwidth estimation to optimize update propagation overhead
- Update tree data structure for fast queries

Full Solution: Outline

- Use *topology server* to coordinate peers
- Random graph model for mesh
- Bandwidth estimation to optimize update propagation overhead
- Update tree data structure for fast queries
- Random back off model

Full Solution: Outline

- Use *topology server* to coordinate peers
- Random graph model for mesh
- Bandwidth estimation to optimize update propagation overhead
- Update tree data structure for fast queries
- Random back off model
- Group size estimation for large groups

Full Solution: Outline

- Use *topology server* to coordinate peers
- Random graph model for mesh
- Bandwidth estimation to optimize update propagation overhead
- Update tree data structure for fast queries
- Random back off model
- Group size estimation for large groups
- WAN and LAN experimental results
- Related work and conclusions

Topology Server

- One global, well known topology server

Topology Server

- One global, well known topology server
  1. Each peer registers with the topology server

Topology Server

- One global, well known topology server
  1. Each peer registers with the topology server
     - Alice: **Register Port 12345 http://www.foo.com/bar.iso**

Topology Server

- One global, well known topology server
  1. Each peer registers with the topology server
     - Alice: **Register Port 12345 http://www.foo.com/bar.iso**
  2. Topology Server returns last ψ peers to register

Topology Server

- One global, well known topology server
  1. Each peer registers with the topology server
     - Alice: **Register Port 12345 http://www.foo.com/bar.iso**
  2. Topology Server returns last ψ peers to register
     - Server: **return:** $\{$**Bob:1111, Cathy:2222, Doug:3333**$\}$, $\psi = 3$

Topology Server

- One global, well known topology server
  1. Each peer registers with the topology server
     - Alice: **Register Port 12345 http://www.foo.com/bar.iso**
  2. Topology Server returns last ψ peers to register
     - Server: **return:** $\{$**Bob:1111, Cathy:2222, Doug:3333**$\}$, $\psi = 3$
- Intuition: last ψ peers are most likely to persist in the system

Mesh and Random Graph Model

- A peer uses seed nodes from topology server to discover other nodes

Mesh and Random Graph Model

- A peer uses seed nodes from topology server to discover other nodes
- Connect to η random peers: called *neighbors*

Mesh and Random Graph Model

- A peer uses seed nodes from topology server to discover other nodes
- Connect to η random peers: called *neighbors*
- Cache node ID/updates for U neighbors

Mesh and Random Graph Model

- A peer uses seed nodes from topology server to discover other nodes
- Connect to η random peers: called *neighbors*
- Cache node ID/updates for U neighbors
  - Incentive: More state $\Rightarrow$ Better Information

Mesh and Random Graph Model

- A peer uses seed nodes from topology server to discover other nodes
- Connect to η random peers: called *neighbors*
- Cache node ID/updates for U neighbors
  - Incentive: More state $\Rightarrow$ Better Information
- Flood update information through the mesh

Mesh and Random Graph Model

- A peer uses seed nodes from topology server to discover other nodes
- Connect to η random peers: called *neighbors*
- Cache node ID/updates for U neighbors
  - Incentive: More state $\Rightarrow$ Better Information
- Flood update information through the mesh
- Periodically disconnect, and reconnect

Mesh and Random Graph Model

- A peer uses seed nodes from topology server to discover other nodes
- Connect to η random peers: called *neighbors*
- Cache node ID/updates for U neighbors
  - Incentive: More state $\Rightarrow$ Better Information
- Flood update information through the mesh
- Periodically disconnect, and reconnect
- Forms random r-regular graph, where $r = \eta$

Bandwidth Estimation

- Simple three state estimation based on passive observation

Bandwidth Estimation

- Simple three state estimation based on passive observation
  - under-utilized,throttle-back,at-capacity

Bandwidth Estimation

- Simple three state estimation based on passive observation
  - under-utilized,throttle-back,at-capacity
- **If** under-utilized, **then** add more peer connections

Bandwidth Estimation

- Simple three state estimation based on passive observation
  - under-utilized,throttle-back,at-capacity
- **If** under-utilized, **then** add more peer connections
- **If** no peer connections to add, **then** increase update rate and number of neighbors

Bandwidth Estimation

- Simple three state estimation based on passive observation
  - under-utilized,throttle-back,at-capacity
- **If** under-utilized, **then** add more peer connections
- **If** no peer connections to add, **then** increase update rate and number of neighbors
- **If** throttle-back first reduce update rate and number of neighbors, **then** remove peer connections

Bandwidth Estimation

- Simple three state estimation based on passive observation
  - under-utilized,throttle-back,at-capacity
- **If** under-utilized, **then** add more peer connections
- **If** no peer connections to add, **then** increase update rate and number of neighbors
- **If** throttle-back first reduce update rate and number of neighbors, **then** remove peer connections
- Update/second changes are AIMD

Update Tree

Possible queries:

Update Tree

Possible queries:

- Which blocks are not in the system?

Update Tree

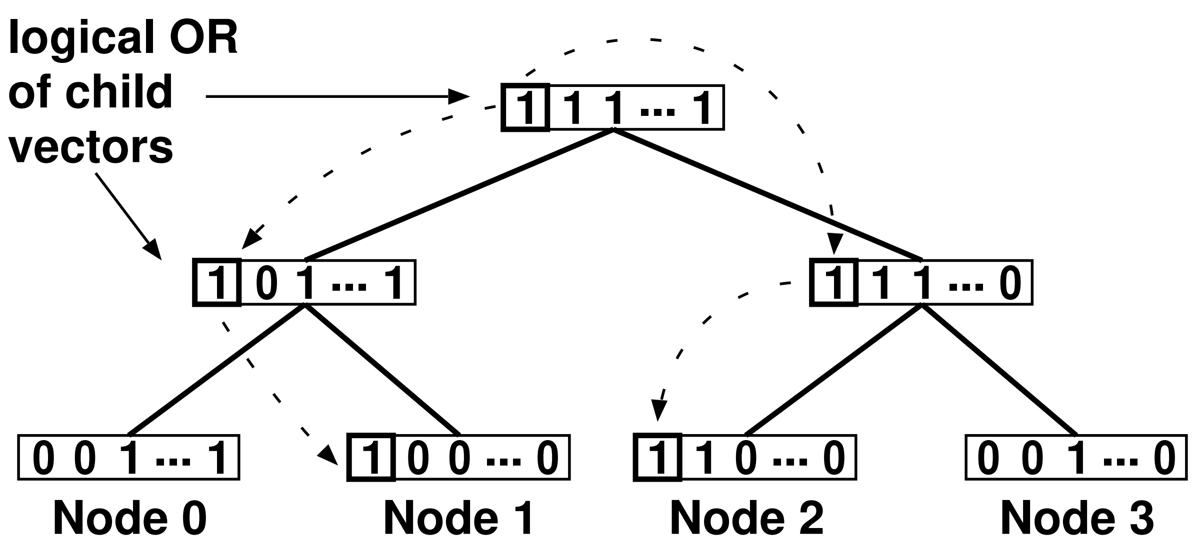
Possible queries:

- Which blocks are not in the system?
- Who has block i ?

Update Tree

Possible queries:

- Which blocks are not in the system?
- Who has block i ?



In implementation, updates are bit vectors.

Random Back off Model

- Last block problem

Random Back off Model

- Last block problem
- Solution: go to the source server with $P(\frac{1}{n})$

Random Back off Model

- Last block problem
- Solution: go to the source server with $P(\frac{1}{n})$
- Discrete values: in practice $P(\frac{3}{n})$ works better

Random Back off Model

- Last block problem
- Solution: go to the source server with $P(\frac{1}{n})$
- Discrete values: in practice $P(\frac{3}{n})$ works better
  - Approximately σ for large n

Random Back off Model

- Last block problem
- Solution: go to the source server with $P(\frac{1}{n})$
- Discrete values: in practice $P(\frac{3}{n})$ works better
  - Approximately σ for large n
- Requires estimation of n,
  i.e. number of peers in system

Random Back off Model

- Last block problem
- Solution: go to the source server with $P(\frac{1}{n})$
- Discrete values: in practice $P(\frac{3}{n})$ works better
  - Approximately σ for large n
- Requires estimation of n,
  i.e. number of peers in system
- Significant performance increase

Group Size Estimation

Group Size Estimation

- Propagate number of neighbors, η, with updates: i.e. our out degree

Group Size Estimation

- Propagate number of neighbors, η, with updates: i.e. our out degree
- Increment hop-count field in updates

Group Size Estimation

- Propagate number of neighbors, η, with updates: i.e. our out degree
- Increment hop-count field in updates
- Use $\bar{\eta}$ for average degree; diameter d=MAX(hop-count)

Group Size Estimation

- Propagate number of neighbors, η, with updates: i.e. our out degree
- Increment hop-count field in updates
- Use $\bar{\eta}$ for average degree; diameter d=MAX(hop-count)
- $n = O((\bar{\eta} - 1)^d)$

Group Size Estimation

- Propagate number of neighbors, η, with updates: i.e. our out degree
- Increment hop-count field in updates
- Use $\bar{\eta}$ for average degree; diameter d=MAX(hop-count)
- $n = O((\bar{\eta} - 1)^d)$
- Loose estimate sufficient

Group Size Estimation

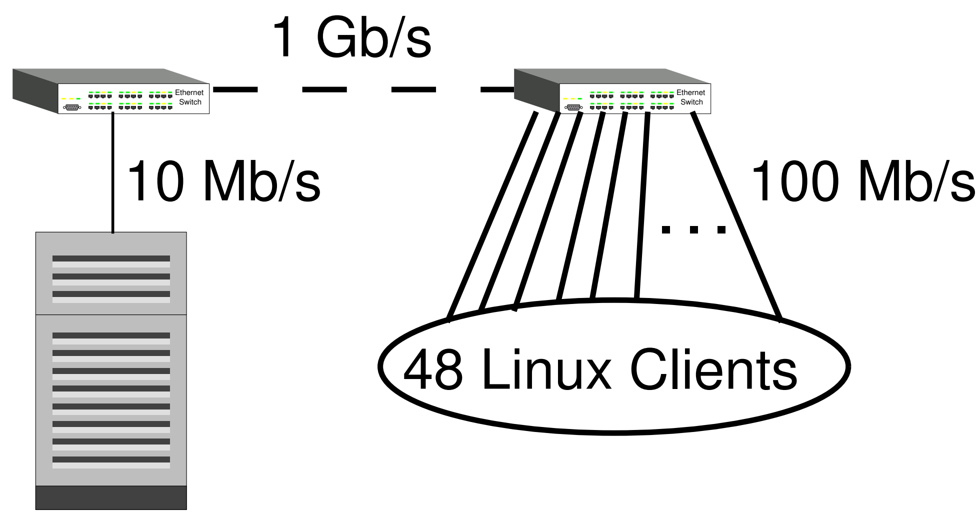
- Propagate number of neighbors, η, with updates: i.e. our out degree
- Increment hop-count field in updates
- Use $\bar{\eta}$ for average degree; diameter d=MAX(hop-count)
- $n = O((\bar{\eta} - 1)^d)$
- Loose estimate sufficient
  - 33.3% error $\Rightarrow \pm$ 1 connection

Group Size Estimation

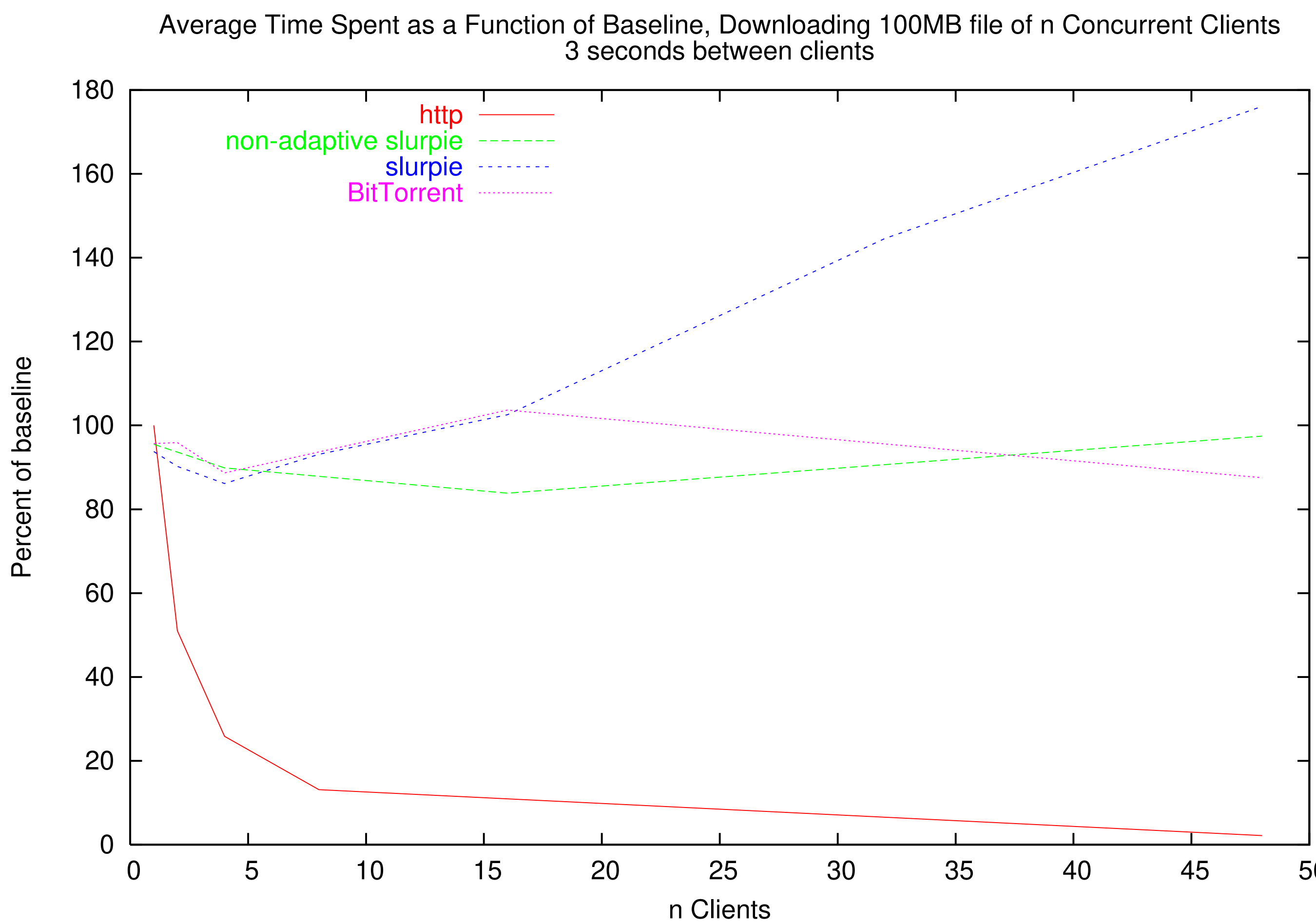
- Propagate number of neighbors, η, with updates: i.e. our out degree
- Increment hop-count field in updates
- Use $\bar{\eta}$ for average degree; diameter d=MAX(hop-count)
- $n = O((\bar{\eta} - 1)^d)$
- Loose estimate sufficient
  - 33.3% error $\Rightarrow \quad \pm$ 1 connection
- Works well in simulation

Experiment Design

- LAN Topology
  - Server on 10Mb/s link
  - 48 GNU/Linux peers
- Planet Lab
  - Same Server
  - 55 GNU/Linux peers, varied geographically

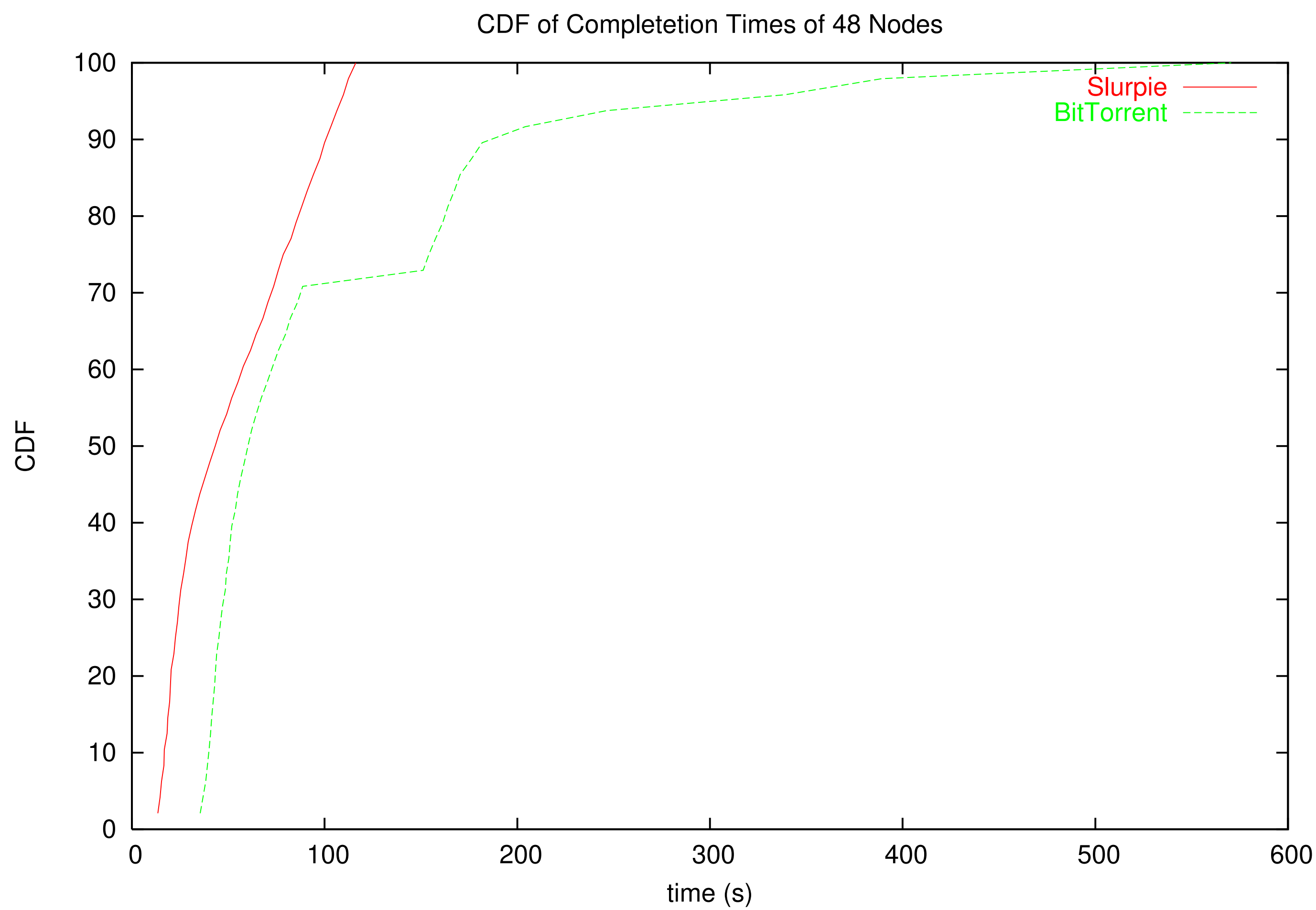


Results - LAN

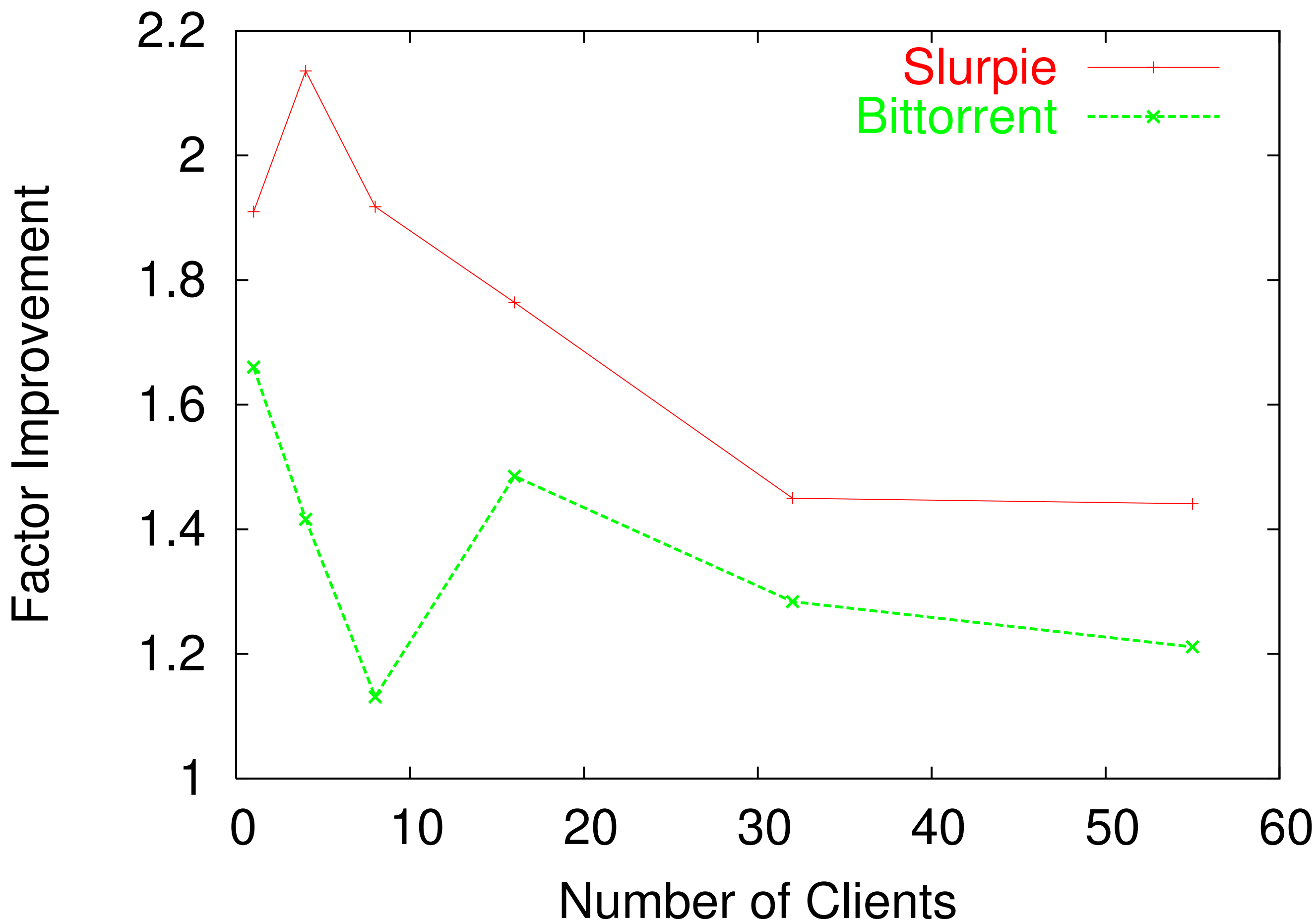


Average Time Spent as a Function of Baseline, Downloading 100MB file of n Concurrent Clients
3 seconds between clients
http
non-adaptive slurpie
slurpie
BitTorrent
Percent of baseline
0
20
40
60
80
100
120
140
160
180
0
5
10
15
20
25
30
35
40
45
50
n Clients

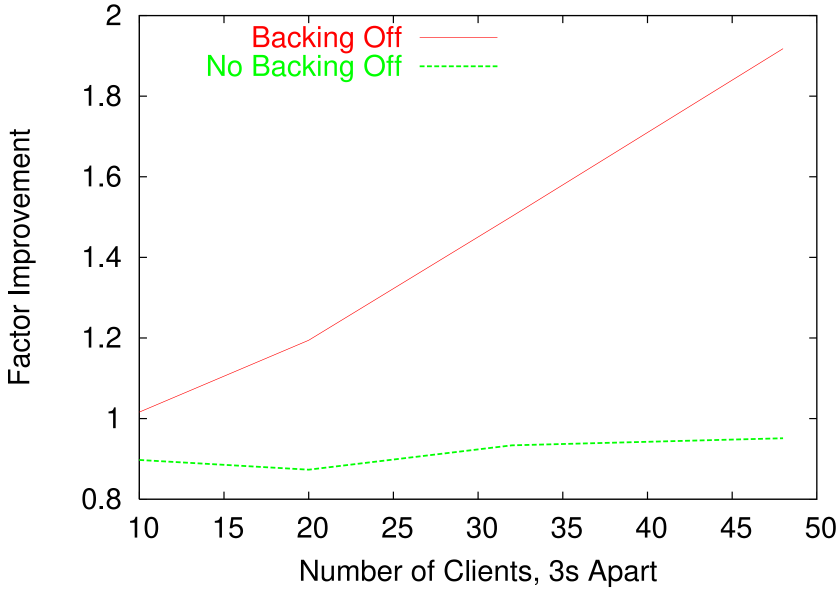
Results - LAN (continued)

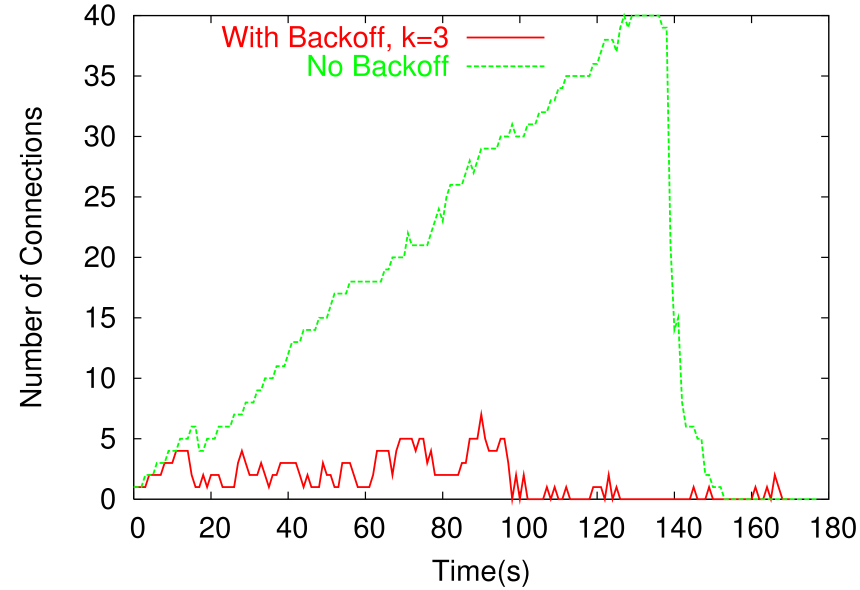


Results - WAN


Slurpie
Bittorrent
Factor Improvement
2.2
2
1.8
1.6
1.4
1.2
1
0
10
20
30
40
50
60
Number of Clients

Effects of Back Off





Related Work

Related Work

- Bittorrent

Related Work

- Bittorrent
  - Server support; “tit-for-tat”

Related Work

- Bittorrent
  - Server support; “tit-for-tat”
- Infrastructure Based: CDNs, Akamai, Web caches

Related Work

- Bittorrent
  - Server support; “tit-for-tat”
- Infrastructure Based: CDNs, Akamai, Web caches
  - Requires a priori knowledge of usage patterns

Related Work

- Bittorrent
  - Server support; “tit-for-tat”
- Infrastructure Based: CDNs, Akamai, Web caches
  - Requires a priori knowledge of usage patterns
- CoopNet: targets small files

Related Work

- Bittorrent
  - Server support; “tit-for-tat”
- Infrastructure Based: CDNs, Akamai, Web caches
  - Requires a priori knowledge of usage patterns
- CoopNet: targets small files
  - Assumes nodes will persist in network

Related Work

- Bittorrent
  - Server support; “tit-for-tat”
- Infrastructure Based: CDNs, Akamai, Web caches
  - Requires a priori knowledge of usage patterns
- CoopNet: targets small files
  - Assumes nodes will persist in network
- IP/End system multicast: DVMRP, Narada, Scribe, NICE

Related Work

- Bittorrent
  - Server support; “tit-for-tat”
- Infrastructure Based: CDNs, Akamai, Web caches
  - Requires a priori knowledge of usage patterns
- CoopNet: targets small files
  - Assumes nodes will persist in network
- IP/End system multicast: DVMRP, Narada, Scribe, NICE
  - Require server side support

Conclusion

Conclusion

- Increasing number of concurrent peers potentially *decreases* download times

Conclusion

- Increasing number of concurrent peers potentially *decreases* download times
- Slurpie significantly out performs non-cooperative and Bittorrent peers

Conclusion

- Increasing number of concurrent peers potentially *decreases* download times
- Slurpie significantly out performs non-cooperative and Bittorrent peers
- Random back off provides significant performance increase

Conclusion

- Increasing number of concurrent peers potentially *decreases* download times
- Slurpie significantly out performs non-cooperative and Bittorrent peers
- Random back off provides significant performance increase
- Linux implementation http://slurpie.sourceforge.net

Future Work

Future Work

- Slurpie proxy

Future Work

- Slurpie proxy
- Better Security

Future Work

- Slurpie proxy
- Better Security
- Take advantage of existing mirrors

Future Work

- Slurpie proxy
- Better Security
- Take advantage of existing mirrors
- Broader testing

Future Work

- Slurpie proxy
- Better Security
- Take advantage of existing mirrors
- Broader testing
- Effects of erasure codes, etc..

Questions?

- ...?