Dynamic Parallel Downloading with Bandwidth Estimation in an Uncooperative environment

Kalyan Boggavarapu
David Manura
CSE 498
Lehigh University

Outline

- Introduction
- Bandwidth estimation (PProbe)
- Dynamic parallel downloading (PDownloader)
- Automatic world-wide name server
- Experimental results and analysis

Contributions:

- Relative bandwidth estimator (PProbe) in Java.
- Dynamic parallel downloader (PDownloader).
- Automated world-wide name server determination.

“Dynamic parallel downloading”

- “Parallel”—To increase the throughput
- “Dynamic”—To download from the best server
- “Uncooperative”—Software need not be installed at the server-side

PProbe

Kalyan Boggavarapu
David Manura
Lehigh University

Problems with the current BW estimators

- Slow
- Some consume lots of bandwidth
- System-dependent
- Firewall/kernel problems
- Requires root permissions to install the tool

Benefits of PProbe Bandwidth Estimator

- Fast—only needs first few packets
- Low bandwidth consumption: needs only the first few packets
- Portable: Java, Platform independent
- No firewall problems because it uses normal TCP packets
- Any user can run it, need not be a root

PProbe Clients

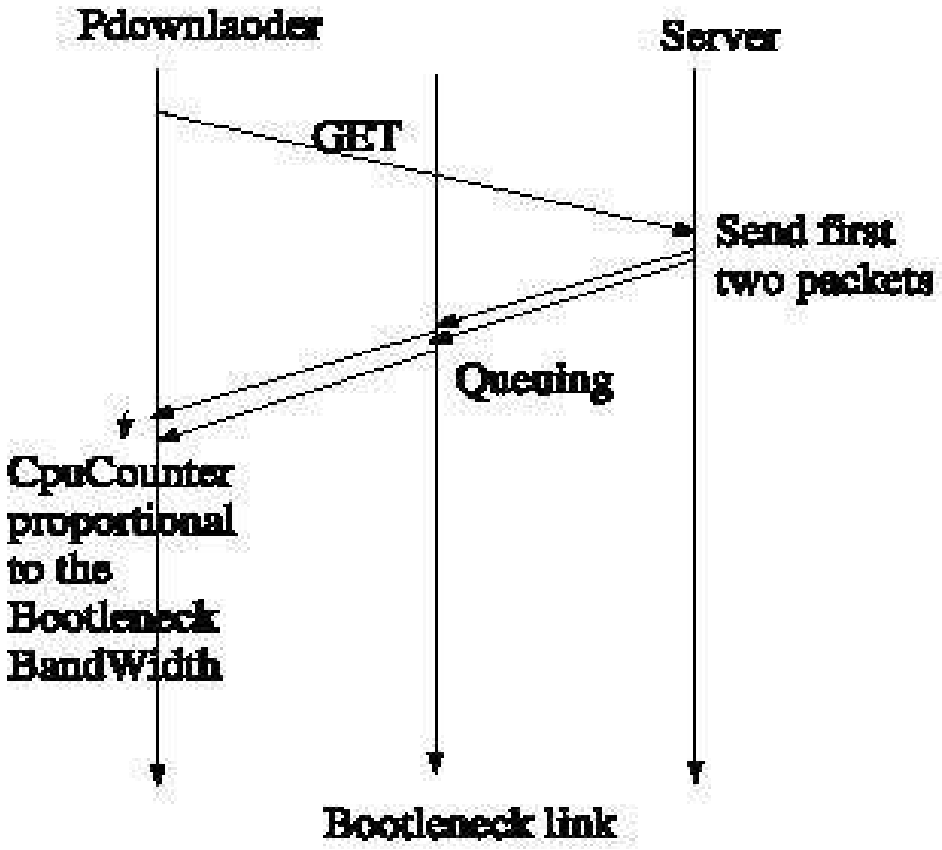
- All Clients:
  - Fast clients:
    - Metric: Bottleneck Bandwidth
  - Fast clients, Servers have the same Bottleneck bandwidth:
    - Metric: Consider the RTT
  - Slow clients:
    - Metric : RTT

Bandwidth Estimator – PProbe Types of implementation

- Applications:
  - Server selection
  - Peer selection in P2P
- Java
  - CPU counter
  - CPU (approx) = CPU cycles
  - Fasttimer
- C
- By default the PProbe uses the Java CPU Counter

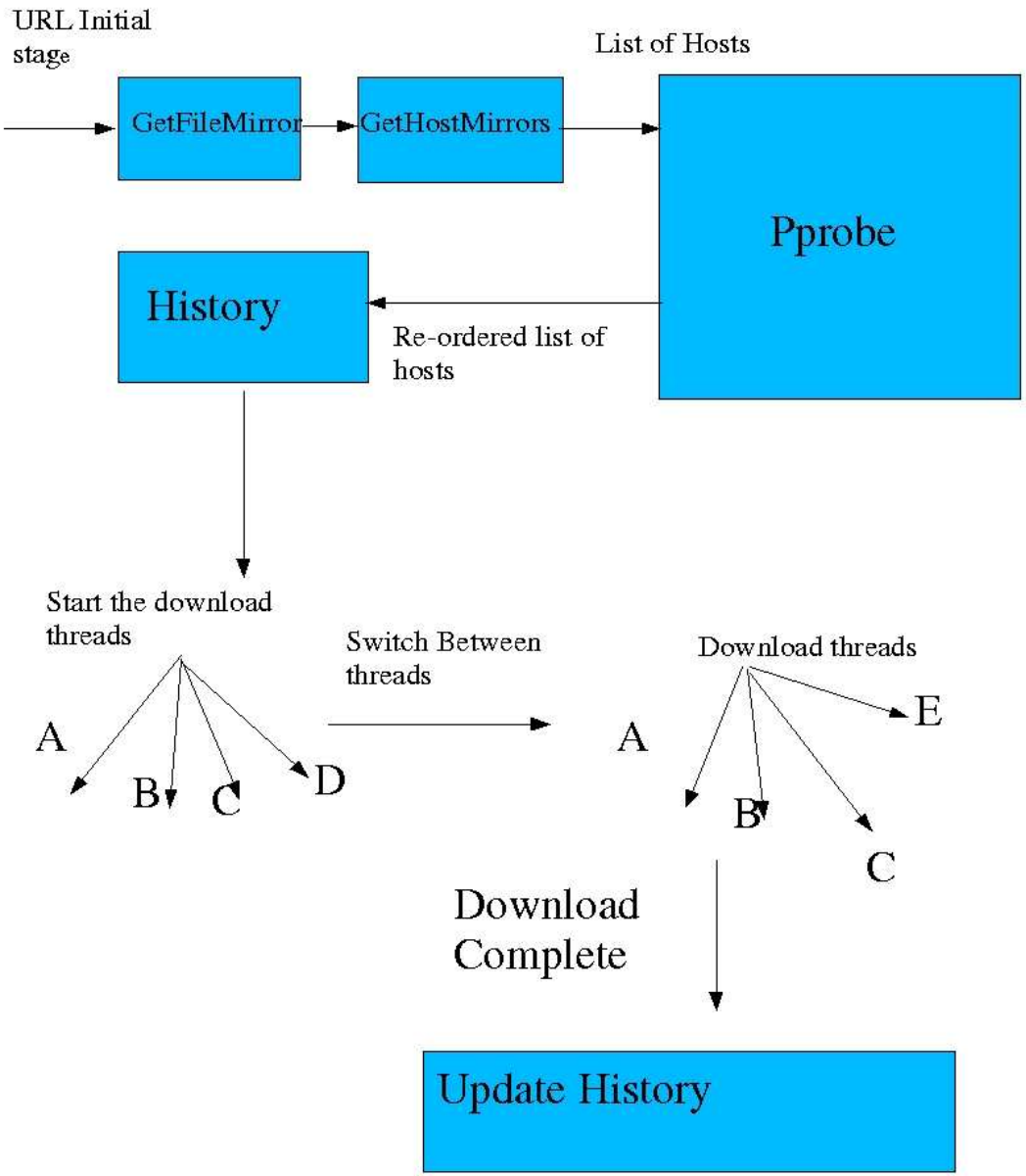
PProbe: Technique

- Packet Pair



Kalyan Boggavarapu
David Manura
Lehigh University

PDownloader



Stages of PDownloader

PDownloader Features

- Fault tolerance
  - Dynamic switching between the best servers
  - If the local servers are down, could use the world-wide name servers list to find out the servers from elsewhere.
- Scalability
  - New mirrors could easily be added
  - Files could be easily be added at new locations
    - Use Google to find the multiple locations of the file
- Clients
  - Fast clients: to get better throughput
  - Slow clients: availability
- Server side: No changes required.
- Client side: could implement the PDownloader

Automatic World-Wide Name Servers

- Purpose
  - To download from world-wide mirror servers
- Technique
  - Get the list of servers from the www.traceroute.org homepage,
  - Parse them and get the name servers
- Infrastructure side: An optional special service could be provided to get the name servers at different locations.
  - We have web page which provides the list of world wide name-servers.
  - http://wume.cse.lehigh.edu/~kcb2/advNet/proj/dns_explore/ips.txt
  - To find the mirror files: two proposed solutions
    - Sol1: Query Google and get the list of locations of the file mirrors
    - Sol2: A special search engine could be hosted like the citeseer, but this one for the files.

Experimental Results and Analysis

Kalyan Boggavarapu
David Manura
Lehigh University

TCP Chunk Traces

C version

Bytes	ΔBytes	$\Delta Time_m$
1460		
2920	1460	0.001135
4380	1460	0.001300
5840	1460	**0.029430**
7300	1460	0.001154
8760	1460	0.001139
10220	1460	0.001494
11680	1460	0.001416
13140	1460	0.001317
14600	1460	**0.026687**
16060	1460	0.001047
17520	1460	0.001339
18980	1460	0.001609
20440	1460	0.001300
21900	1460	0.001275
23360	1460	0.001352
24820	1460	0.001151

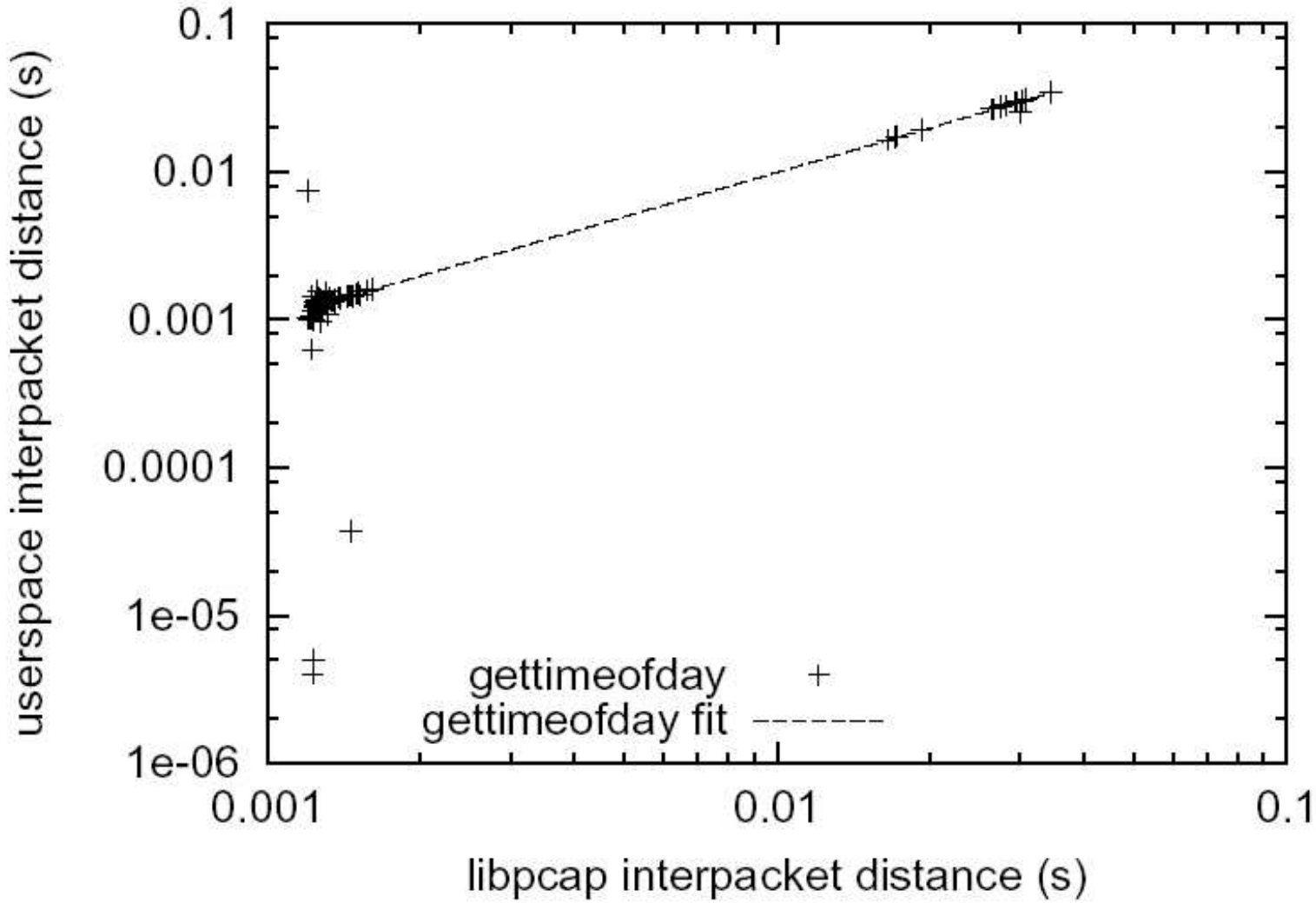
Java version

Bytes	ΔBytes	$\Delta Time_u$	$\Delta Time_m$	$\Delta Time_c$
4380				
8760	**4380**	0.027657	0.028	0.020378
10220	1460	0.001312	0.001	0.001872
11680	1460	0.001382	0.002	0.001819
13140	1460	0.001296	0.001	0.001819
14600	1460	0.025469	0.025	0.027849
20440	**4380**	0.004606	0.005	0.000787
23360	1460	0.001497	0.001	0.002213
24820	1460	0.001725	0.002	0.001958
26280	1460	0.001038	0.001	0.001042
27740	1460	0.001555	0.001	0.001319
30660	**2920**	0.004999	0.005	0.001564
31184	**524**	0.016136	0.017	0.017324

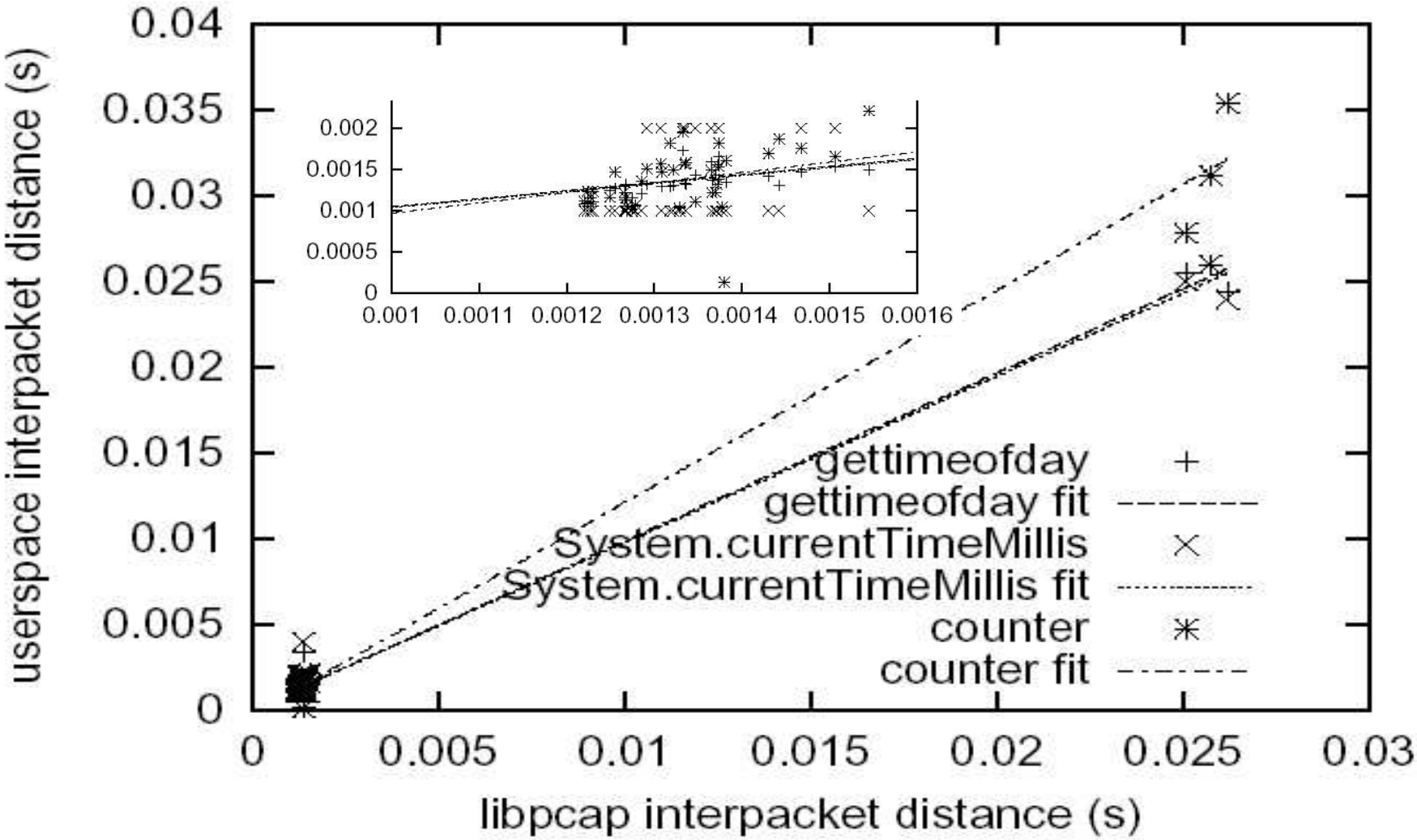
On download of http://www.yahoo.com.
Note: RTT = 30 ms (ICMP)

Kalyan Boggavarapu
David Hartley
Lehigh University

C version: chunk-packet correlation



Java version: chunk-packet correlation



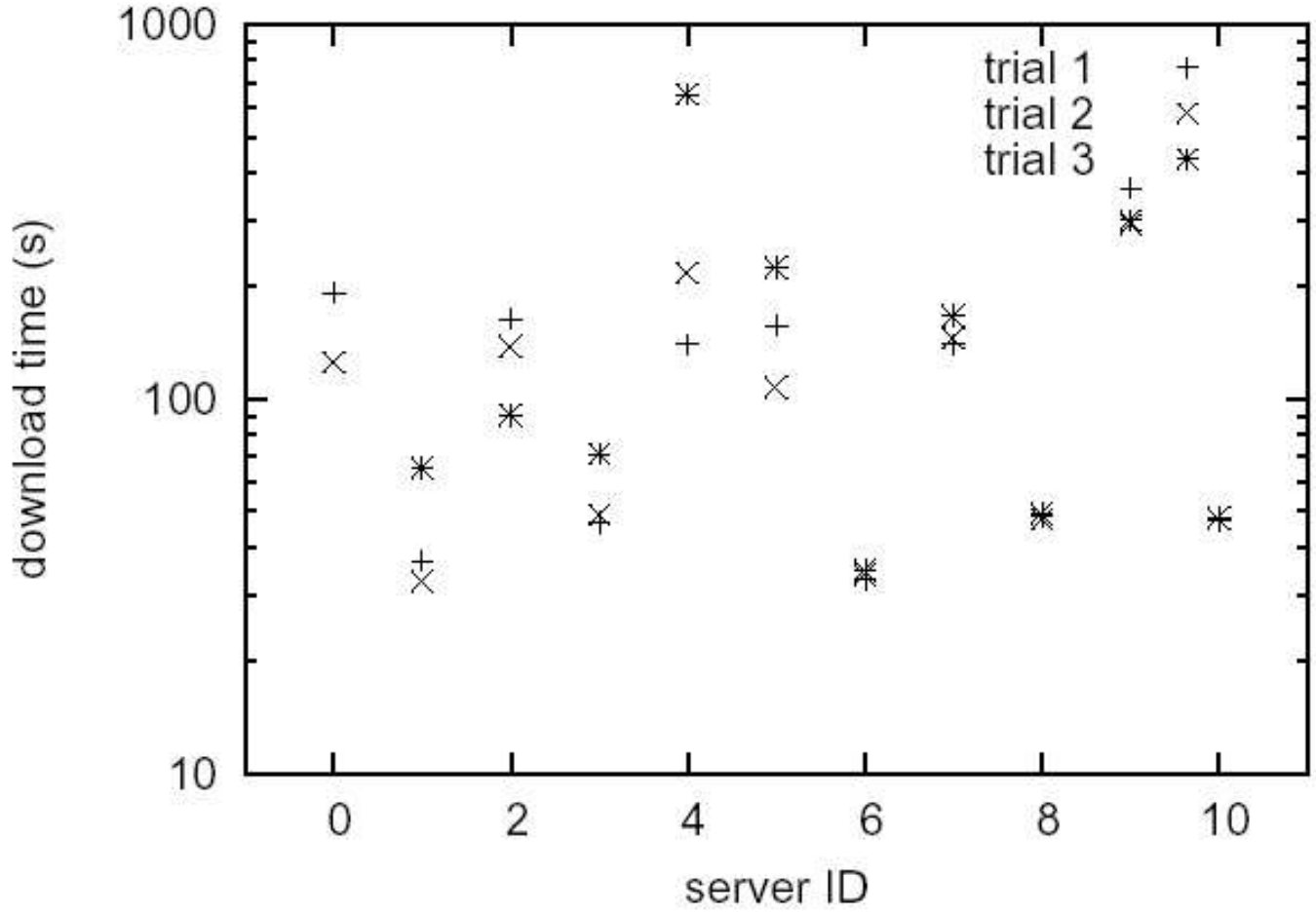
Parallel Downloading Experiment

- Servers used:

0: aurolinux.mit.edu
1: cudlug.cudenver.edu
2: debian-cd.rutgers.edu
3: debian.midco.net
4: debian.oregonstate.edu
5: ftp.keystealth.org
6: ftp.lug.udel.edu
7: ftp.rutgers.edu
8: linux.csua.berkeley.edu
9: mirror.csit.fsu.edu
10: mirrors.usc.edu

First 20 MB of debian-30r1-i386-binary-1.iso

Single Downloading

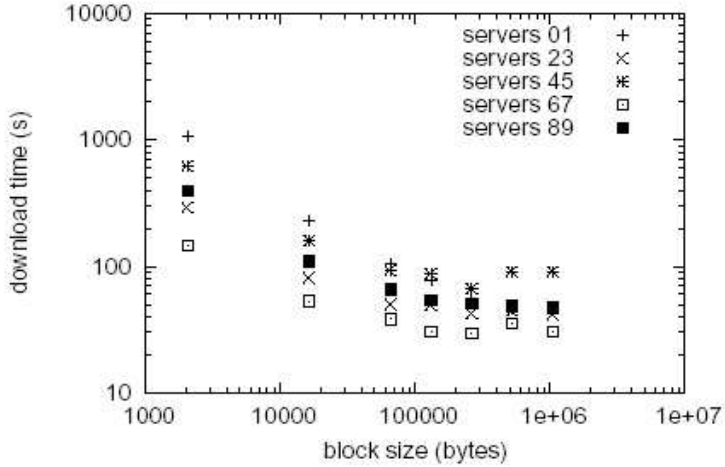


20 MB requests to Debian servers.

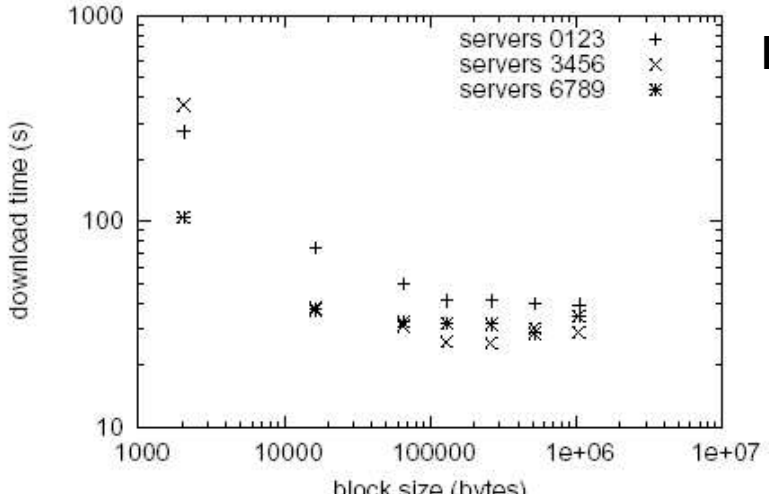
Kalyan Boggavarapu
David Manura
Lehigh University

Parallel Downloading

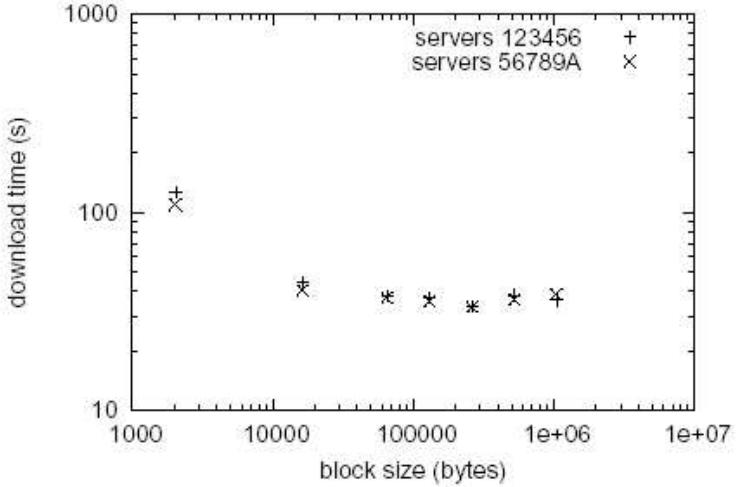
N=2



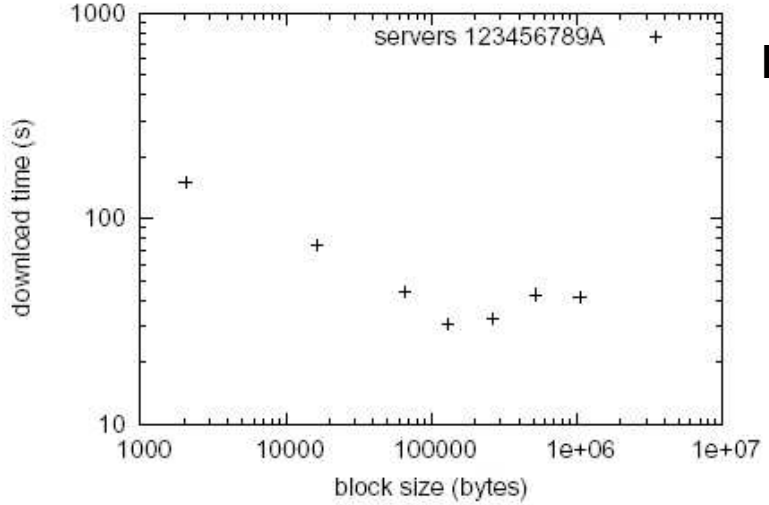
N=4



N=6



N=10



ı Bog
David Manura
Lehigh University

20 MB requests to Debian servers.

Web Server Support

Web Mirror Set	web100.com	mirc.com
number of servers	100	40
contactable servers	90	24
server type		
Apache	39	20
IIS	22	2
Netscape	20	1
features		
RR + PC	20	20
RR + !PC	5	2
!RR + PC	37	0
!RR + !PC	28	2
auto disConc time(s)		
0-1	33	4
1-30	34	18
30-inf	23	2
HTTP version		
1.1	84	22
1.0	6	2

Conclusions