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

Ki
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](http://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.
  - 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
## TCP Chunk Traces

<table>
<thead>
<tr>
<th>C version</th>
<th>Java version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes</td>
<td>ΔBytes</td>
</tr>
<tr>
<td>1460</td>
<td>2920</td>
</tr>
<tr>
<td>4380</td>
<td>1460</td>
</tr>
<tr>
<td>5840</td>
<td>1460</td>
</tr>
<tr>
<td>7300</td>
<td>1460</td>
</tr>
<tr>
<td>8760</td>
<td>1460</td>
</tr>
<tr>
<td>10220</td>
<td>1460</td>
</tr>
<tr>
<td>11680</td>
<td>1460</td>
</tr>
<tr>
<td>16060</td>
<td>1460</td>
</tr>
<tr>
<td>17520</td>
<td>1460</td>
</tr>
<tr>
<td>18980</td>
<td>1460</td>
</tr>
<tr>
<td>20440</td>
<td>1460</td>
</tr>
<tr>
<td>21900</td>
<td>1460</td>
</tr>
<tr>
<td>23360</td>
<td>1460</td>
</tr>
<tr>
<td>24820</td>
<td>1460</td>
</tr>
</tbody>
</table>

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

Kalyan Boggavarapu
David Rotea
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.
Parallel Downloading

N=2

N=6

N=4

N=10

20 MB requests to Debian servers.
## Web Server Support

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