Towards Estimating Available Bandwidth

Ningning Hu
Peter Steenkiste

Departments of Computer Science and Electrical and Computer Engineering
Carnegie Mellon University
Vrije Universiteit, Amsterdam, March 17, 2006
Why Measuring Available Bandwidth?

- Picking servers: selecting the server with the highest throughput.
- Multi-homing: selecting what ISP to use to a particular destination.
- Overlay networking: selecting links and routes that provide the highest throughput.
- End-to-end diagnostics: detecting poor connectivity and identifying the link or ISP that is responsible.

Must be light weight!
Outline

- Packet pair background.
- PTR: measuring available bandwidth.
- Pathneck: identifying the bottleneck link.
- BRoute: estimating available bandwidth.
- Conclusion.
Available Bandwidth

- The residual bandwidth on the bottleneck link

Available bandwidth

Background traffic

Bottleneck link capacity

100Mbps  10Mbps  100Mbps
Measurement Techniques

- **Passive measurement**: record the statistics of ongoing network transmission.
  - SNMP, NetFlow
  - SPAND

- **Active measurement**: inject probing packets into the network.
  - Traceroute, ping
  - Bprobe, nettimer, pathchar, cprobe, pathload
Packet Pair Probing: Intuition

- Gap between the two packets of a packet pair can change as it travels over a link
  - Interleaving with competing traffic
  - Effect of low-capacity paths

- Packet trains similarly mingle with the background traffic
Packet Pair Probing for Available Bandwidth

- Effects on the bottleneck link dominate the packet gap changes along the path.
  - Other links do not affect the gap much

- Comparing the gap at the source and destination may yield information about the bottleneck and thus the end-to-end available bandwidth.

- But it is not that simple ..

![Diagram showing packet gaps along a path with a bottleneck link highlighted.](image-url)
A Simple Experiment

- Send sequence of trains with increasing gap and record average output gap
- Turning point: packet train exactly fills the unused bandwidth on the bottleneck link
  - Faster: flooding the link
  - Slower: leaving unused capacity
- Analytical model backs up this intuition
Estimating the Available Bandwidth

1. Find the turning point by probing with trains with different input gaps until input = output

2. Estimate the available bandwidth:
   - By using the packet train rate (PTR)
     \[ B_a = \frac{\sum p_b}{\sum g_o} \]
   - or -
   - By estimating the competing bandwidth observed by different pairs in the train (IGI)
     \[ B_a = B_o - \left( \frac{\sum g_o^+}{\sum g_o} \right) \times B_o \]
Experiment on 13 Internet paths, with different network properties

Refer to our paper for detail properties of the Internet paths
## Measurement Time

<table>
<thead>
<tr>
<th>Path ID</th>
<th>( IGI/PTR ) (s)</th>
<th>Pathload (s)</th>
<th>Ratio(( \frac{Pathload}{IGI/PTR} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(5%, median, 95%)</td>
<td>(5%, median, 95%)</td>
<td>median</td>
</tr>
<tr>
<td>1</td>
<td>(1.60, 2.05, 6.27)</td>
<td>(14.98, 30.56, 31.03)</td>
<td>13.22</td>
</tr>
<tr>
<td>2</td>
<td>(0.58, 0.73, 1.56)</td>
<td>(13.67, 15.37, 31.81)</td>
<td>20.86</td>
</tr>
<tr>
<td>3</td>
<td>(0.11, 0.11, 0.18)</td>
<td>(7.55, 13.17, 14.91)</td>
<td>99.78</td>
</tr>
<tr>
<td>4</td>
<td>(0.49, 0.52, 0.52)</td>
<td>(11.78, 12.26, 12.76)</td>
<td>23.48</td>
</tr>
<tr>
<td>5</td>
<td>(0.78, 0.80, 0.83)</td>
<td>(15.58, 15.86, 16.55)</td>
<td>19.75</td>
</tr>
<tr>
<td>6</td>
<td>(0.62, 0.80, 1.20)</td>
<td>(49.07, 56.18, 62.24)</td>
<td>70.08</td>
</tr>
<tr>
<td>7</td>
<td>(0.51, 0.51, 0.67)</td>
<td>(14.01, 22.40, 28.51)</td>
<td>45.94</td>
</tr>
<tr>
<td>8</td>
<td>(1.01, 1.02, 1.27)</td>
<td>(27.57, 31.51, 47.62)</td>
<td>27.80</td>
</tr>
<tr>
<td>9</td>
<td>(0.24, 0.30, 0.30)</td>
<td>(15.35, 16.14, 27.66)</td>
<td>65.81</td>
</tr>
<tr>
<td>10</td>
<td>(1.27, 1.27, 1.50)</td>
<td>(20.95, 21.04, 21.77)</td>
<td>16.50</td>
</tr>
<tr>
<td>11</td>
<td>(1.03, 1.10, 2.03)</td>
<td>(19.97, 25.78, 38.52)</td>
<td>23.45</td>
</tr>
<tr>
<td>12</td>
<td>(2.17, 2.32, 3.60)</td>
<td>(19.24, 21.54, 42.00)</td>
<td>9.20</td>
</tr>
<tr>
<td>13</td>
<td>(1.10, 1.11, 1.13)</td>
<td>(12.24, 12.76, 47.22)</td>
<td>11.24</td>
</tr>
</tbody>
</table>

| Geometric Mean | 26.39 |
## Related Work

<table>
<thead>
<tr>
<th>What is measured</th>
<th>How to measure</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uniform probing</td>
<td>Non-uniform probing</td>
</tr>
</tbody>
</table>
| Available BW (rate) | Pathload, PTR, TOPP | pathChirp | Does not use link capacity  
|                   |                |             | Measures trains |
| Competing BW (gap) | IGI            | Spruce      | Uses link capacity  
|                   |                |             | Measures gaps |
| Differences:      | Small averaging interval | Longer averaging interval |          |
Outline

- Packet pair background.
- PTR: measuring available bandwidth.
- Pathneck: identifying the bottleneck link.
- BRoute: estimating available bandwidth.
- Conclusion.
Yes, but Where Is the Bottleneck?

- Can we measure the available bandwidth on every link along the path?
- Requires us to measure the gap not only at the source and destination but at every router on the path.

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
</table>

- Idea: combine a PTR train with measurement packets that probe the routers.
Recursive Packet Train (RPT)
Transmission of RPT
### Example: 139.82.0.1

```
[root@azure src-all]# ./pathneck -f 139.82.0.1
1080697381.041055 139.82.0.1 500 60 0
```

<table>
<thead>
<tr>
<th>N</th>
<th>Path Length</th>
<th>Path Length</th>
<th>IP Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0.773</td>
<td>2871</td>
<td>128.2.223.254</td>
<td>GIGROUTER.NET.CS.CMU.EDU</td>
</tr>
<tr>
<td>01</td>
<td>0.770</td>
<td>2744</td>
<td>128.2.0.12</td>
<td>CORE0-VL1000.GW.CMU.NET</td>
</tr>
<tr>
<td>02</td>
<td>4.886</td>
<td>1627</td>
<td>128.2.33.233</td>
<td>HYPER-VL502.GW.CMU.NET</td>
</tr>
<tr>
<td>03</td>
<td>1.135</td>
<td>2748</td>
<td>192.88.115.185</td>
<td>bar-cmu-ge-4-0-0-1.psc.net</td>
</tr>
<tr>
<td>04</td>
<td>1.008</td>
<td>2872</td>
<td>192.88.115.18</td>
<td>beast-bar-g4-0-1.psc.net</td>
</tr>
<tr>
<td>05</td>
<td>10.244</td>
<td>15738</td>
<td>192.88.115.124</td>
<td>abilene-psc.abilene.ucaid.edu</td>
</tr>
<tr>
<td>06</td>
<td>26.106</td>
<td>2995</td>
<td>198.32.8.65</td>
<td>atlang-washng.abilene.ucaid.edu</td>
</tr>
<tr>
<td>07</td>
<td>44.839</td>
<td>2871</td>
<td>198.32.252.253</td>
<td>abilene-oc3.ampath.net</td>
</tr>
<tr>
<td>08</td>
<td>149.269</td>
<td>8368</td>
<td>198.32.252.238</td>
<td>rnp.ampath.net</td>
</tr>
<tr>
<td>09</td>
<td>147.519</td>
<td>9865</td>
<td>200.143.254.94</td>
<td>rj-fast4_1.bb3.rnp.br</td>
</tr>
<tr>
<td>10</td>
<td>148.261</td>
<td>9740</td>
<td>200.143.254.77</td>
<td>200.143.254.77</td>
</tr>
<tr>
<td>11</td>
<td>157.876</td>
<td>83448</td>
<td>200.20.94.81</td>
<td>rederio-atm-puc.rederio.br</td>
</tr>
<tr>
<td>12</td>
<td>324.145</td>
<td>24989</td>
<td>200.20.91.250</td>
<td>200.20.91.250</td>
</tr>
<tr>
<td>13</td>
<td>298.034</td>
<td>51093</td>
<td>139.82.59.65</td>
<td>139.82.59.65</td>
</tr>
<tr>
<td>14</td>
<td>317.538</td>
<td>31082</td>
<td>139.82.59.3</td>
<td>b40.rdc.puc-rio.br</td>
</tr>
<tr>
<td>15</td>
<td>321.010</td>
<td>26731</td>
<td>139.82.183.60</td>
<td>mec-fw.mec.puc-rio.br</td>
</tr>
</tbody>
</table>
Detecting the Bottleneck

Train rate < Available BW
Train rate > Available BW

Gap

choke points

bottleneck point

Hop count
But Not So Fast

- **Bursty traffic**: can cause “random” changes in the train length.
- **Bursty traffic on the reverse path** can affect the measurement packets.
  - Adds more noise to the data
- **ICNP packet generation time on routers** can be long and variable.
  - Adds even more noise
Pathneck: Algorithm Overview

1. Probe the same dst N times (N=10)

2. Adjust outliers in the gap traces

4. Detect choke point candidates for each probing
   » Compute a confidence measure for each detected choke point

5. Identify final choke point set based ranking across probings. The last choke point is the bottleneck link
Validation

● Controlled experiments on Emulab.
  » Single bottleneck: always identifies correct bottleneck (R6- R7)
  » Two bottleneck (R2- R3 added): identifies both in correct order 75% of the time; identifies first bottleneck only 25% of the time
  » Reverse path traffic: only R6- R7 is labeled as choke point more than 50% of the time, i.e. it is identified as only valid choke point

● “Real” network experiments on Abilene.
  » Link utilization statistics are available for most links
  » Pathneck results match the statistics
Pathneck Properties

- Does not need help from receiver
  - Relies on ICMP responses only
  - Convenient for large scale bandwidth studies!

- Relatively low overhead
  - Load packets: $60 \times 500B = 30KB$
  - Measurement packets: $2 \times 30 \times 60B = 3.6KB$
  - Instantaneous load (22.4 TCP pkts) $= = TCP$ slow start

- Directly comparable packet measurements
  - All the gap measurements are from the same packet train

- Hidden horizon: cannot “see” beyond bottleneck link.
  - E.g. some later links may have less bandwidth than earlier choke points
Pathneck Challenges

- **Firewall**
  - Most firewall drops any non-ping/traceroute probing packets → load packets will be dropped

- **ICMP packet generation time**
  - Most routers have 200 – 500 us ICMP generation delay, with a periodic delay as large as 1ms

- **Reverse path congestion can affect the packet gap measurements.**
  - Can affect measurements either way
  - Can generate “hill” and “valley” points

- **Need small variant to measure the gap on the last link**
  - ICMP echo response
Related Work

C **Pathchar**: measure transmission time on links.
   - Incremental probing of successive hops
   - Need lots of probes to filter our noise

C **Cartouche**: target probing packets on particular sub-paths (Boston U.).
   - Use “pacing” to skip early part of path
   - Use separate measurement packets to skip the tail end

A **Tailgating**: target partial path by combining load and measurement packets (Rice).
   - Load packets expire and measurement packets report delay

A **Bfind**: gradually increase UPD load on path and measure queue build up using traceroute (CMU).
Outline

- Packet pair background.
- PTR: measuring available bandwidth.
- Pathneck: identifying the bottleneck link.
- BRoute: estimating available bandwidth.
- Conclusion.
Large Scale Bandwidth Estimation

- Estimate the available bandwidth for the $N^2$ paths in a $N$-node system when $N$ is very large (> 1000)

- Applications:
  - P2P applications
  - ISP customer monitoring
  - Server selection

- Difficulty: measurement overhead
  - The best so far: 100KB per path $\rightarrow$ 2GB for 150-node system
Source of Inspiration: Delay Estimation with O(N) Cost

- Insight: Scalable measurement should be based on node-specific information which is independent of node number
  - For RTT: the coordinate

- Is this possible for available bandwidth?
Where Should We Focus?

- Most of the bottlenecks in the Internet are at the edges of the path.
  - Access links or access networks

- How can we efficiently monitor the network edge?
IP-Level Source/Sink Tree

- IP-level source/sink tree is the graph composed of all the first-X-hop upstream or downstream routes of the end nodes.

For a path (a, c), we need:

1. Obtain the source tree (for a) and sink tree (for c)
2. Label each branch with bandwidth information
3. Find out the source & sink segment that a path uses
Identify Source and Sink Trees + Measure Bandwidths

- Use traceroute to identify trees and Pathneck to estimate bandwidth of the tree branches
- Trees plus bandwidth info constitute “coordinates”

Requires Landmarks!
Mapping End Segments

- Use the common-AS to map the end-segments

Algorithm:
1. Infer AS-level source/sink trees from IP-level source/sink trees
2. Identify the shared ASes
3. Use the shared AS in the lowest tier as the Common-AS
4. Use Common-AS to map back to IP-level tree branches

AS-Level Source/Sink Tree

Common-AS
Validation

● Are AS-level source/sink trees really trees?
  » 90% of the 190 AS-Level source trees have a proximity over 0.95 and over 99% are above 0.88 (sink trees similar)

● Common-AS identification accuracy
  » 97% accuracy for 15,383 AS-path tests

● Unique mapping from Common-AS to IP-level tree branches
  » 85% map onto an unique IP-level tree branch
  » 13% map onto 2 IP-level tree branches
  » We consider both (so as to cover 98% of cases)

● Small case study on 160 Planetlab nodes
  » 80% of estimations have available bandwidth measurement error within 50%
  
  Note: Planetlab is really too small for BRoute to be effective
Conclusion

● Packet train techniques can estimate available bandwidth at the turning point.
  » Packets in train interact with competing traffic
  » Probing packets do not flood the link

● The bottleneck link can be identified by combining probing train with measurement packets.
  » Measures train length at every router, identifying choke points where available bandwidth is reduced
  » Last choke point is the bottleneck for the path

● BRoute can possibly reduce cost of estimating available bandwidth to O(n) for large systems.
  » Leverage source and sink tree topology in the network edge
  » Use AS-level topology information to derive routes
References


● Exploiting Internet Route Sharing for Large Scale Available Bandwidth Estimation, Ningning Hu and Peter Steenkiste, 2005 Internet Measurement conference (IMC 2005).

● Tools available at: http://www.cs.cmu.edu/~hnn