Optimizing TCP Forwarder Performance

Oliver Spatscheck, Jørgen S. Hansen, Student Member, IEEE,
John H. Hartman, Member, IEEE, and
Larry L. Peterson, Senior Member, IEEE

2000 IEEE

Shing-Guo Chang
Outline

• TCP Forwarder
• Connection splicing
• Connection scout
• Test
• Conclusion
TCP Forwarder

• Firewall
  - ex: when accept connections, need authentication
TCP Forwarder

- **HTTP proxy**
- Content-based switch can optimize the connection by packet contents
Connection splicing

- **Proxy mode**
  - Control mode: firewall authentication
  - Forward mode: HTTP proxy forward packet from server

- **Forward mode**
  - It only adjust headers or cache data
  - Where is overhead?
Connection splicing

TCP splice
Connection splicing

• Forwarding
  - Port number: two TCP connection may be other
  - Seq. number: it will be different, because TCP initialize
  - ACK number: it is dissimilar by the direction
  - Checksum: modify header have to adjust
Connection splicing
Connection splicing

- Output.DstPort = RemotePort(B)
- Output.SrcPort = LocalPort(B)
- Output.SeqNum = Input.SeqNum + SeqNumOffset(A->B)
- Output.Ack = Input.Ack
Connection splicing

• When Splice, we should check the receive queue in unoptimized TCP
  - New packet put in wait queue
  - Sometimes have window size problem

• When uns splice
  - Check all packet are acknowledged in optimized TCP
  - Forward mode to control mode
Unsplice is Asked
Next Seq = SA, SB

TCP Segments
Come In from A

SEQ

Yes (∨ New Data)

No (¬ Old Data)

SEQ > SA

Forward Data

Wait Queue

ACK

YES

ACK >= SB

Forward Ack
If Necessary

No (¬ Old Ack)

Drain
Wait Queue

A is Unspliced
Connection scout

- Scout define path abstraction that encapsulates data
- Each path have two important define
  - A sequence of code module are applied to data
  - Represent the entity is scheduled for execution
Test

- All host are 200MHz CPU with 256KB cache
- 128MB ram
- Digital Fast EtherWORKS PCI 10/100
- Linux 2.0.30
Test

Back-to-back

Network interface latency
<table>
<thead>
<tr>
<th>Configuration</th>
<th>1-byte TCP segments</th>
<th>1460-byte TCP segments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Processing time</td>
<td>Speedup</td>
</tr>
<tr>
<td></td>
<td>(µsecs)</td>
<td></td>
</tr>
<tr>
<td>Scout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-path</td>
<td>68.5</td>
<td>–</td>
</tr>
<tr>
<td>1-path</td>
<td>66.1</td>
<td>1.04</td>
</tr>
<tr>
<td>FWD</td>
<td>39.0</td>
<td>1.76</td>
</tr>
<tr>
<td>IP/FWD</td>
<td>24.0</td>
<td>2.85</td>
</tr>
<tr>
<td>IP router</td>
<td>22.4</td>
<td>3.06</td>
</tr>
<tr>
<td>Linux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIS Firewall</td>
<td>83.9</td>
<td>–</td>
</tr>
<tr>
<td>Filtering IP router</td>
<td>27.5</td>
<td>3.05</td>
</tr>
<tr>
<td>IP router</td>
<td>25.5</td>
<td>3.29</td>
</tr>
</tbody>
</table>
Conclusion

• We describes connection splicing can be applied to TCP forwarder
• In the future, we are interest in SSL-secured HTTP connections