Scalable TCP: Improving Performance in HighSpeed Wide Area Networks

PFLDnet 2003
CERN, Geneva

Tom Kelly
ctk21@cam.ac.uk

CERN
and
Laboratory for Communication Engineering
University of Cambridge
Poor performance of TCP in high bandwidth wide area networks due to TCP congestion control algorithm

- for each ack in a RTT without loss:
  \[
  cwnd_r \rightarrow cwnd_r + \frac{1}{cwnd}
  \]
- for each window experiencing loss:
  \[
  cwnd_r \rightarrow cwnd_r - \frac{1}{2}cwnd_r
  \]

<table>
<thead>
<tr>
<th>Throughput</th>
<th>Window</th>
<th>Loss recovery time</th>
<th>Supporting loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10Mbps</td>
<td>170pkts</td>
<td>17s</td>
<td>$5.4 \times 10^{-5}$</td>
</tr>
<tr>
<td>100Mbps</td>
<td>1700pkts</td>
<td>2mins 50s</td>
<td>$5.4 \times 10^{-7}$</td>
</tr>
<tr>
<td>1Gbps</td>
<td>17000pkts</td>
<td>28mins</td>
<td>$5.4 \times 10^{-9}$</td>
</tr>
<tr>
<td>10Gbps</td>
<td>170000pkts</td>
<td>4hrs 43mins</td>
<td>$5.4 \times 10^{-11}$</td>
</tr>
</tbody>
</table>

Characteristics of a 200ms, 1500 MTU TCP connection
Changing congestion control - aims and assumptions

- Make effective use of high bandwidth links
- Changes need to be robust in a wide variety of networks and traffic conditions
  - L2 switches, bugs, packet corruption, reordering and jitter
- Do not adversely damage existing network traffic
- Do not require manual tuning to achieve reasonable performance
  - 80% of maximal performance for 95% of the people is fine
The generalised Scalable TCP algorithm

Let \( a \) and \( b \) be constants and \( cwnd \) be the congestion window

- for each ack in a RTT without loss:
  \( cwnd \rightarrow cwnd + a \)
- for each window experiencing loss:
  \( cwnd \rightarrow cwnd - b \times cwnd \)

Loss recovery times for RTT 200ms and MTU 1500bytes

- Scalable TCP: \( \frac{\log(1-b)}{\log(1+a)} \) RTTs
  e.g. if \( a = 0.01, b = 0.125 \) then it is about 2.7s
- Traditional: at 50Mbps about 1min 38s, at 500Mbps about 27min 47s!
The Scalable TCP algorithm

Scalable TCP: Improving Performance in High-Speed Wide Area Networks – p.5/14
Choose a legacy window size, $lwnd$

When $cwnd > lwnd$ use the Scalable TCP algorithm

When $cwnd \leq lwnd$ use traditional TCP algorithm

Same argument used in the HighSpeed TCP proposal

Fixing $lwnd$, fixes the ratio $\frac{a}{b}$
Theorem (Vinnicombe): The generalised Scalable TCP algorithm is locally stable about equilibrium, if

\[ a < \frac{p_j(\hat{y}_j)}{\hat{y}_j p'_j(\hat{y}_j)} \quad \forall j \in J \]

where \( \hat{y}_j \) is the equilibrium rate at each link, \( p_j(y) \) is the probability of loss at link \( j \) for an arrival rate \( y \), and \( J \) is the set of all links.

With appropriate buffer sizes or AQM stability can be ensured.
Variance and Convergence

<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
<th>Rate CoV</th>
<th>Loss recovery time</th>
<th>Rate halving time</th>
<th>Rate doubling time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{2}{50}$</td>
<td>0.50</td>
<td>$17.7T_r$ (3.54s)</td>
<td>$T_r$ (0.20s)</td>
<td>$17.7T_r$ (3.54s)</td>
</tr>
<tr>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{50}$</td>
<td>0.35</td>
<td>$14.5T_r$ (2.91s)</td>
<td>$2.41T_r$ (0.48s)</td>
<td>$35T_r$ (7.00s)</td>
</tr>
<tr>
<td>$\frac{1}{8}$</td>
<td>$\frac{1}{100}$</td>
<td>0.25</td>
<td>$13.4T_r$ (2.68s)</td>
<td>$5.19T_r$ (1.04s)</td>
<td>$69.7T_r$ (13.9s)</td>
</tr>
<tr>
<td>$\frac{1}{16}$</td>
<td>$\frac{1}{200}$</td>
<td>0.18</td>
<td>$12.9T_r$ (2.59s)</td>
<td>$10.7T_r$ (2.15s)</td>
<td>$139T_r$ (27.8s)</td>
</tr>
</tbody>
</table>
lwnd = 16, a = 0.01, and b = 0.125 represents a good trade off of concerns

Patch against Linux 2.4.19 implements Scalable TCP algorithm

- Linux already implements reordering detection, SACK, and rate halving

Some driver details (bugs?) fixed for Gbps operations
- DatATAG 2.4Gbps link and minimal buffers (2048/40)
- Flows transfer 2 gigabytes and start again for 1200s

<table>
<thead>
<tr>
<th>Number of flows</th>
<th>2.4.19 TCP</th>
<th>2.4.19 TCP &amp; giga-bit device buffer</th>
<th>Scalable TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>16</td>
<td>44</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>39</td>
<td>93</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
<td>60</td>
<td>135</td>
</tr>
<tr>
<td>8</td>
<td>47</td>
<td>86</td>
<td>140</td>
</tr>
<tr>
<td>16</td>
<td>66</td>
<td>106</td>
<td>142</td>
</tr>
</tbody>
</table>
### Web traffic results

- DataTAG 2.4Gbps link and minimal buffers (2048/40)
- 4 bulk concurrent flows across 2 machines for 1200s
- 4200 concurrent web users across 3 machines

<table>
<thead>
<tr>
<th>Type of bulk transfer users</th>
<th>Web traffic transferred</th>
<th>2 Gigabyte transfers completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>No bulk transfers</td>
<td>65GB</td>
<td>n/a</td>
</tr>
<tr>
<td>TCP in 2.4.19</td>
<td>65GB</td>
<td>36</td>
</tr>
<tr>
<td>TCP in 2.4.19 &amp; giga-bit device buffers</td>
<td>65GB</td>
<td>58</td>
</tr>
<tr>
<td>Scalable TCP</td>
<td>65GB</td>
<td>96</td>
</tr>
</tbody>
</table>
Conclusion

- Strong theoretical framework behind the algorithm
- Offers an easy evolution from the traditional TCP AMID scheme
- Freely available working code
  http://www-lce.eng.cam.ac.uk/~ctk21/scalable
Where from here?

- Correcting RTT bias in throughput allocation; methods similar to the parameter scaling used in previous ECN work
- Better code efficiency to improve robustness and performance of implementation
- AQM and ECN evolutions that can give extra performance in some scenarios
More at

http://www-lce.eng.cam.ac.uk/~ctk21/scalable