HighSpeed TCP and Quick-Start for Fast Long-Distance Networks:

Sally Floyd
Topics:

• HighSpeed TCP.
  S. Floyd, HighSpeed TCP for Large Congestion Windows
  RFC 3649
  URL http://www.icir.org/floyd/hstcp.html

  S. Floyd, Limited Slow-Start for TCP with Large Congestion Windows
  RFC 3742
  URL http://www.icir.org/floyd/hstcp.html

• Quick-Start.
  A. Jain, S. Floyd, M. Allman, and P. Soralahti,
  Quick-Start for TCP and IP.
  draft-amit-quick-start-03.txt
  URL: http://www.icir.org/floyd/quickstart.html
The Problem: TCP for High-Bandwidth-Delay-Product Networks

* Sustaining high congestion windows:

A Standard TCP connection with:
- 1500-byte packets;
- a 100 ms round-trip time;
- a steady-state throughput of 10 Gbps;

would require:
- an average congestion window of 83,333 segments;
- and at most one drop (or mark) every 5,000,000,000 packets
  (or equivalently, at most one drop every 1 2/3 hours).

This is not realistic.
Is this a pressing problem in practice?

- **Nope.** In practice, users do one of the following:
  - Open up $N$ parallel TCP connections; or
  - Use MulTCP (roughly like an aggregate of $N$ virtual TCP connections).

- **However, we can do better:**
  - Better flexibility (no $N$ to configure);
  - Better scaling (with a range of bandwidths, numbers of flows);
  - Better slow-start behavior;
  - Competing more fairly with current TCP (for environments where TCP is able to use the available bandwidth).
What is HighSpeed TCP:

• Just like Standard TCP when cwnd is low.

• More aggressive than Standard TCP when cwnd is high.
  – Uses a modified TCP response function.

• HighSpeed TCP can be thought of as behaving as an aggregate of $N$ TCP connections at higher congestion windows.

• Joint work with Sylvia Ratnasamy and Scott Shenker, additional contributions from Evandro de Souza, Deb Agarwal, Tom Dunigan.
HighSpeed TCP: the modified response function.

Regular TCP (S = 1.22/p^0.5)
Highspeed TCP (S = 0.15/p^0.82)

Sending Rate S (in pkts/RTT)
Loss Rate P

(10^{-7}, 83000)
(15^{-3}, 31)
HighSpeed TCP: Relative fairness.

Relative Fairness \(\frac{0.11}{p^{0.32}}\)

[Graph showing the relationship between Loss Rate \(p\) and the Highspeed TCP / Regular TCP Sending Rates]
**HighSpeed TCP: The Gory Details:**

<table>
<thead>
<tr>
<th>w</th>
<th>a(w)</th>
<th>b(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>118</td>
<td>2</td>
<td>0.44</td>
</tr>
<tr>
<td>221</td>
<td>3</td>
<td>0.41</td>
</tr>
<tr>
<td>347</td>
<td>4</td>
<td>0.38</td>
</tr>
<tr>
<td>495</td>
<td>5</td>
<td>0.37</td>
</tr>
<tr>
<td>663</td>
<td>6</td>
<td>0.35</td>
</tr>
<tr>
<td>851</td>
<td>7</td>
<td>0.34</td>
</tr>
<tr>
<td>1058</td>
<td>8</td>
<td>0.33</td>
</tr>
<tr>
<td>1284</td>
<td>9</td>
<td>0.32</td>
</tr>
<tr>
<td>1529</td>
<td>10</td>
<td>0.31</td>
</tr>
<tr>
<td>1793</td>
<td>11</td>
<td>0.30</td>
</tr>
<tr>
<td>2076</td>
<td>12</td>
<td>0.29</td>
</tr>
<tr>
<td>2378</td>
<td>13</td>
<td>0.28</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84035</td>
<td>71</td>
<td>0.10</td>
</tr>
</tbody>
</table>
HighSpeed TCP (red) compared to Standard TCP (green).
HighSpeed TCP in a Drop-Tail Environment?

- Drop-Tail queues: a packet is dropped when the (fixed) buffer overflows.

- Active Queue Management: a packet is dropped before buffer overflow. E.g. RED, where the average queue size is monitored.

- In a Drop-Tail environment:
  Assume that TCP increases its sending rate by \( P \) packets per RTT. Then \( P \) packets are likely to be dropped for each congestion event for that connection.
Relative Fairness with RED queue management:

Simulations from Evandro de Souza.
Relative Fairness with Drop-Tail queue management:

Simulations from Evandro de Souza.
Conclusions:

My own view is that this approach is the fundamentally correct path:
- given backwards compatibility and incremental deployment.

More results are on the HighSpeed TCP web page.
- http://www.icir.org/floyd/hstcp.html
- Simulations from Evandro de Souza and Deb Agarwal (LBNL).
- Experimental results from Tom Dunigan (ORNL).
- Experimental results from Brian Tierney (LBNL).
- Experimental results from Les Cottrel (SLAC).
- Experimental results from Tom Kelly on Scalable TCP.
HighSpeed TCP requires Limited Slow-Start:

- Slow-starting up to a window of 83,000 packets doesn’t work well.
  - Tens of thousands of packets dropped from one window of data.
  - Slow recovery for the TCP connection.

- The answer: Limited Slow-Start
  - Agent/TCP set max_ssthresh_ N
  - During the initial slow-start, increase the congestion window by at most N packets in one RTT.
Tests from Tom Dunigan:

This shows Limited Slow-Start, but not HighSpeed TCP.
Other small changes for high congestion windows:

* More robust performance in paths with reordering:
  Wait for more than three duplicate acknowledgments before retransmitting a packet.

* Recover more smoothly when a retransmitted packet is dropped.
Additional Problems:

• Starting up with high congestion windows?

• Making prompt use of newly-available bandwidth?
What is QuickStart?

• In an IP option in the SYN packet, the sender’s desired sending rate:
  – Routers on the path decrement a TTL counter,
  – and decrease the allowed sending rate, if necessary.

• The receiver sends feedback to the sender in the SYN/ACK packet:
  – The sender knows if all routers on the path participated.
  – The sender has an RTT measurement.
  – The sender can set the initial congestion window.
  – The TCP sender continues with AIMD using normal methods.

• From an initial proposal by Amit Jain
The Quick-Start Request Option for IPv4

<table>
<thead>
<tr>
<th>Option</th>
<th>Length=4</th>
<th>QS TTL</th>
<th>Rate Request</th>
</tr>
</thead>
</table>

- Explicit feedback from all of the routers along the path would be required.
- This option will only be approved by routers that are significantly underutilized.
- No per-flow state is kept at the router.
Quick-Start in the NS Simulator:

- Added to NS by Srikanth Sundarrajan.
Questions:

- Would the benefits of Quick-Start be worth the added complexity? 
  – SYN and SYN/ACK packets would not take the fast path in routers.
- Is there a compelling need to add some form of feedback from routers such as this (in addition to ECN)?
- Is there a compelling need for more fine-grained or more frequent feedback, or more congestion-related feedback, than Quick-Start?
- Are there other mechanisms that would be preferable to Quick-Start?
Architectural sub-themes favoring incremental deployment:

- A goal of incremental deployment in the current Internet.
- Steps must go in the fundamentally correct, long-term direction, not be short-term hacks.
- Robustness in heterogeneous environments valued over efficiency of performance in well-defined environments.
- A preference for simple mechanisms, but a skepticism towards simple traffic and topology models.
- Learning from actual deployment is an invaluable step.
- The Internet will continue to be decentralized and fast-changing.
Extra slides:
The Limited Slow-Start pseudocode:

* 

For each arriving ACK in slow-start:

If (cwnd <= max_ssthresh)
   cwnd += MSS;
else
   K = 2 * cwnd/max_ssthresh ;
   cwnd += MSS/K ;