TCP Variants Performance over Tiny Buffer High-speed Network

Cheng cui
### TCP congestion control

**TABLE I**
**Classification of important TCP congestion control variants**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Detection</th>
<th>Probing/Backoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reno [RFC 2581]</td>
<td>Loss</td>
<td>AI/MD</td>
</tr>
<tr>
<td>Coupled [1]</td>
<td>Loss</td>
<td>AI/MD in each path</td>
</tr>
<tr>
<td>HSTCP [RFC 3649]</td>
<td>Loss</td>
<td>Convex AI/MD</td>
</tr>
<tr>
<td>CUBIC [2]</td>
<td>Loss</td>
<td>Concave-convex AI/MD</td>
</tr>
<tr>
<td>Scalable TCP [3]</td>
<td>Loss</td>
<td>Multiplicative incr./MD</td>
</tr>
<tr>
<td>H-TCP [4]</td>
<td>Loss</td>
<td>Convex AI/MD</td>
</tr>
<tr>
<td>Westwood+ [5]</td>
<td>Loss</td>
<td>AI/bandwidth estimation</td>
</tr>
<tr>
<td>FAST [7]</td>
<td>Delay</td>
<td>Function of RTT</td>
</tr>
<tr>
<td>Hybla [8]</td>
<td>Hybrid</td>
<td>AI/MD</td>
</tr>
<tr>
<td>Compound [9]</td>
<td>Hybrid</td>
<td>AI+delay component/ MD</td>
</tr>
<tr>
<td>Illinois [10]</td>
<td>Hybrid</td>
<td>Concave AI/MD</td>
</tr>
</tbody>
</table>

Legend: AI: Additive Increase  MD: Multiplicative Decrease
packet pacing

• packet pacing can be applied both in OSI model:
  – Transport layer [18]
  – Link layer [19]

• techniques impact packet pacing
  – TCP-offloading (sender)
  – large-receive-offloading (receiver)
  – interrupt coalescing (general)

• should we consider disabling all of the impacts?
Background Traffic

• The burstiness of network traffic is roughly categorized into two classes:
  – “long-term” or “elephants”
  – “short-term”, or “mice” [20].

• short-lived flows are very aggressive
  – within a few rtt or dozens of packets, instant throughput could be huge
TCP loss synchronization

![Graphs showing synchronization]

Fig. 1. Synthetic demonstration of complete synchronization (left) and complete loss de-synchronization (right)

- **single flow synchronization rate:**
  \[
  SR_i = \frac{N_i^w}{T} = \frac{1}{T} \sum_{k=1}^{T} (l_{i,k} \times weight)
  \]

- **total flows synchronization rate:**
  \[
  \overline{SR} = \frac{n}{\sum_{i=1}^{n} \frac{1}{SR_i}}
  \]
tracking TCP traces

Algorithm III.1 optimized TCP multi-flow probe

\[
\text{while each TCP packet of a specific flow do}
\]
\[
\text{if (current_cwnd ≠ previous_cwnd) then}
\]
\[
\text{record flow_id, timestamp, cwnd, srtt, tcp_ca_state}
\]
\[
\text{if (current_tcp_ca_state == TCP_CA_Recovery) ∧ (previous_tcp_ca_state == TCP_CA_Open || TCP_CA_Disorder) then}
\]
\[
\text{loss_flag ← 1}
\]
\[
\text{else}
\]
\[
\text{loss_flag ← 0}
\]
\[
\text{end if}
\]
\[
\text{end if}
\]
\[
\text{previous_cwnd ← current_cwnd}
\]
\[
\text{previous_tcp_ca_state ← current_tcp_ca_state}
\]
\[
\text{end while}
\]

TABLE II
COMPARISON OF CPU USAGE AND STORAGE BETWEEN TCPDUMP/LIBPCAP METHOD AND OUR KERNEL METHOD OF SNIFFING 10 TCP FLOWS

<table>
<thead>
<tr>
<th>Method</th>
<th>Network</th>
<th>Packet Size</th>
<th>CPU Usage</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcpdump</td>
<td>1 Gbps</td>
<td>1500 bytes</td>
<td>14%</td>
<td>1GB</td>
</tr>
<tr>
<td>our method</td>
<td>1 Gbps</td>
<td>1500 bytes</td>
<td>6%</td>
<td>22MB</td>
</tr>
<tr>
<td>tcpdump</td>
<td>10 Gbps</td>
<td>4000 bytes</td>
<td>45%</td>
<td>8GB</td>
</tr>
<tr>
<td>our method</td>
<td>10 Gbps</td>
<td>4000 bytes</td>
<td>10%</td>
<td>125MB</td>
</tr>
</tbody>
</table>

- TCP multi-flows tracing algorithm
- implemented as `TCP_multiflow_probe` Kernel Module
- TCPDump first 100 bytes of packet
Queues in Linux kernel

incoming packets → \textit{rx\_ring} → \textit{backlog} → \textit{IP processing and forwarding} → \textit{qdisc} → \textit{tx\_ring} → outgoing packets

- 48 packets
- 2 packets
- 2 packets
- 48 packets
Testbed: 1Gbps network in Utah Emulab

- 1Gbps network, MTU == 1500 bytes
- server1 60 ms delay, server2 120 ms delay, background traffic 90 ms delay
- All buffers (input & output) sized at 50 packets \( \approx 0.24\% BDP_{max} \)
  - \( BDP_{max} == 1\text{Gbps} \times 250\text{ms} == 20833 \) packets
- each link is confirmed at line speed
Background Traffic

Pareto distribution with shape parameter alpha 1.5 (long-tail), and average file size 100.8KB. These values are realistic, based on comparisons with actual packet traces from time to time.
Test Scenarios

• Each long flow server
  – 1, 3, 6, 12, 24, 48 flows

• Total long flows
  – 2, 6, 12, 24, 48, 96 flows

• Background traffic
  – 16.5Mbps of short-lived TCP traffic by Harpoon (Pareto distribution)
  – 30Mbps of UDP traffic,
  – 16.5Mbps + 30Mbps ≈ 5% link capability

• Popular TCP variants
  – Reno, HSTCP, CUBIC, and Coupled (mptcp)

• Trials
  – without background traffic, all experiments repeat 4 times
  – with background traffic, all experiments repeat 6 times
Test Metrics

• average link utilization
  – (total goodputs) / ( 1 – protocol overhead)

• RTT fairness
  – Jain’s fairness index between throughputs of two long-lived traffic servers

• TCP loss synchronization rate
  – overall loss synchronization ratio between all long-lived TCP flows
Average Link Utilization
(no packet pacing, no background traffic)
RTT Fairness
(no packet pacing, no background traffic)

Graph showing the Jain's fairness index as a function of the number of connections, with Reno, HSTCP, CUBIC, and a baseline represented by different markers and colors.
Loss Synchronization (no Packet Pacing, no Background Traffic)
Average Link Utilization
(with Packet Pacing, no background traffic)
RTT Fairness
(disable offloading, no background traffic)
Loss Synchronization
(with packet Pacing, no Background Traffic)
Average Link Utilization
(with packet pacing, with background traffic)
Average Link Utilization -- Reno vs. Coupled-mptcp (with packet pacing, with background traffic)
RTT Fairness -- Reno vs. Coupled-mptcp (with packet pacing, with background traffic)
Loss Synchronization
(Pacing, with Background Traffic)
RENO: 2 connections cwnd behavior (no Packet Pacing, no Background Traffic)
RENO: 96 connections cwnd behavior (no Packet Pacing, no Background Traffic)
HSTCP: 2 connections cwnd behavior (no Packet Pacing, no Background Traffic)
HSTCP: 96 connections cwnd behavior (no Packet Pacing, no Background Traffic)
CUBIC: 2 connections cwnd behavior (no Packet Pacing, no Background Traffic)
CUBIC: 96 connections cwnd behavior
(no Packet Pacing, no Background Traffic)
RENO: 2 connections cwnd behavior (with Packet Pacing, no Background Traffic)
RENO: 96 connections cwnd behavior (with Packet Pacing, no Background Traffic)
HSTCP: 2 connections cwnd behavior (with Packet Pacing, no Background Traffic)
HSTCP: 96 connections cwnd behavior (with Packet Pacing, no Background Traffic)
CUBIC: 2 connections cwnd behavior (with Packet Pacing, no Background Traffic)
CUBIC: 96 connections cwnd behavior (with Packet Pacing, no Background Traffic)
Coupled-2subflows(mptcp):
2 connections cwnd behavior
(with Packet Pacing, no Background Traffic)
Coupled-8subflows(mptcp):
2 connections cwnd behavior
(with Packet Pacing, no Background Traffic)
Coupled-30 subflows (mptcp):
2 connections cwnd behavior
(with Packet Pacing, no Background Traffic)
RENO: 2 connections cwnd behavior, test5 (with Packet Pacing, with Background Traffic)
RENO: 96 connections cwnd behavior
(with Packet Pacing, with Background Traffic)
HSTCP: 2 connections cwnd behavior, test5 (with Packet Pacing, with Background Traffic)
HSTCP: 96 connections cwnd behavior (with Packet Pacing, with Background Traffic)
CUBIC: 2 connections cwnd behavior (with Packet Pacing, with Background Traffic)
CUBIC: 96 connections cwnd behavior (with Packet Pacing, with Background Traffic)
Coupled-2subflows(mptcp):
2 connections cwnd behavior, test4
(with Packet Pacing, with Background Traffic)
Coupled-8subflows(mptcp):
2 connections cwnd behavior, test4
(with Packet Pacing, with Background Traffic)
Coupled-30subflows(mptcp):
2 connections cwnd behavior, test3
(with Packet Pacing, with Background Traffic)
Coupled-30 subflows (mptcp):
2 connections cwnd behavior in detail
(with Packet Pacing, with Background Traffic)