Extending TCP Congestion Control Algorithm for Multimedia Streaming

Midori Kato
Keio University
Motivation

• **Smooth congestion control** is important for multimedia service (e.g. VoIP and video streaming)
  • Many apps use TCP to traverse NAT, firewall etc.
  • TCP congestion control is not appropriate (i.e. sawtooth throughput behavior)
TFWC (TCP-Friendly Window Control) ※[1]

• Smooth congestion control
• TCP-Friendly
  • share the equal bandwidth with competing flows in the link
• Window based variant of TFRC (DCCP’s congestion control algorithm)
  • Implementable in TCP
• I extend TFWC for multimedia services

How smooth should flows be for multimedia services?

- depends on apps
- Smoothness and throughput are trade-off

Interactive apps
- cannot buffer much data.
- Smoothness is very important
- VoIP
- Video conferencing

Non-interactive apps
- can buffer some data.
- Perfect smoothness is not required, but require high throughput
- On-demand video
- Audio streaming
Trade-off between smoothness and throughput

Quickly react to available bandwidth

More bandwidth becomes available

TCP

TFWC

TFWC is smooth, but too conservative throughput

throughput

\( t \)
• Problem

• Apps have only two choices:
  • very flappy and high throughput (TCP),
    or
  • very smooth and low throughput (TFWC)
Contribution

• My work addresses this problem
• Tunable TFWC (TTFWC) allows users and apps to optimize the balance between smoothness and throughput
  • We define the tuning parameter to change throughput behavior
TTFWC throughput behavior

More bandwidth becomes available

Non-interactive apps (large parameter value)
Interactive apps (small parameter value)
TFWC window behavior

Point 1: Window reduction after a packet loss is small.

Point 2: TFWC does not increase its window size for a certain period.

Point 3: TFWC window increases slower than TCP.

TFWC window behavior is much smoother than TCP.
Why TFWC gets lower throughput than TCP

Total amount of data between TCP and TFWC is almost same while TFWC keeps the window size.

TFWC window behavior results in lower throughput than TCP.
Modification 1: TTFWC adds the base window size

Modification 2: TTFWC increases the window size more quickly than TFWC

Both modifications are based on the tuning parameter
Evaluation: experiment setup

- Implement TTFWC as TCP extension in Linux kernel 3.0
  - Low-speed network: $B = 10$ Mbps
  - High-speed network: $B = 250$ Mbps

**dummynet**: tune delay and bandwidth of the link

B = bottleneck bandwidth
RTT = 16 ms – 320 ms
Throughput and TCP-Friendliness[2/3]

Results in the low-speed network

TCP throughput

TTFWC throughput

4% throughput increasing

TCP-Friendly line
Throughput and TCP-Friendliness[3/3]

Results in the high-speed network

<table>
<thead>
<tr>
<th></th>
<th>TFWC</th>
<th>TTFWC(0.4)</th>
<th>TTFWC(0.9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth usage</td>
<td>1.80</td>
<td>1.78</td>
<td>1.93</td>
</tr>
<tr>
<td>TCP throughput</td>
<td>1.77</td>
<td>1.67</td>
<td>1.60</td>
</tr>
</tbody>
</table>

7% throughput increasing
CPU Processing Time[1/2]

- Measure two main congestion control functions with ftrace
  - Next window calculation and window size recalculation procedures
  - Ftrace: enable to trace procedures inside the Linux kernel from user space

* Next window calculation procedure

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Average time (µsec)</th>
<th>Standard deviation (µsec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENO</td>
<td>0.240</td>
<td>0.015</td>
</tr>
<tr>
<td>TTFWC (0.0)</td>
<td>1.770</td>
<td>0.271</td>
</tr>
<tr>
<td>TTFWC (0.4)</td>
<td>1.951</td>
<td>0.380</td>
</tr>
<tr>
<td>TTFWC (0.9)</td>
<td>1.975</td>
<td>0.400</td>
</tr>
<tr>
<td>CUBIC</td>
<td>1.771</td>
<td>1.069</td>
</tr>
</tbody>
</table>

Reasonable overhead comparing to current system
## CPU Processing Time[2/2]

* Window size recalculation procedure

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Average time (µsec)</th>
<th>Standard deviation (µsec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENO</td>
<td>0.225</td>
<td>0.00</td>
</tr>
<tr>
<td>TTFWC (0.0)</td>
<td>2.200</td>
<td>0.052</td>
</tr>
<tr>
<td>TTFWC (0.4)</td>
<td>2.258</td>
<td>0.508</td>
</tr>
<tr>
<td>TTFWC (0.9)</td>
<td>2.159</td>
<td>0.439</td>
</tr>
<tr>
<td>CUBIC</td>
<td>0.189</td>
<td>0.00</td>
</tr>
</tbody>
</table>

High overhead comparing to others. But this function is not frequently called.
Conclusion

• Different types of multimedia services require a variety of throughput behavior

• TTFWC allows users and apps to optimize the balance between smoothness and high throughput

• On-going work
  • Give me advice to improve performance of sshthresh re-calculation, \textit{ttfwc\_ssthresh()}
  • See \url{http://www.ht.sfc.keio.ac.jp/~katoon/ttfwc/}
Thank you for your attention
Metric of smoothness

- I define smoothness as the responsiveness of its throughput when the available bandwidth changes.

- Expectation of throughput oscillation
  - Small parameter (0.0 - 0.5)
    - Square of tuning parameter value
  - Large parameter (0.5 – 1.0)
    - Tuning parameter values
Throughput and TCP-Friendliness

• In the high-speed network,
  • TTFWC throughput influences RTT values
    • Short delay: TTFWC throughput > TCP throughput
    • Long delay: TTFWC throughput ≈ TCP throughput

Due to TCP sawtooth shape window behavior, TCP generates idle bandwidth in the high-speed network in comparison in the low-speed network.
Smoothness [1/4]

- NS-2 simulation result
- TTFWC throughput behavior with eight flows in the low-speed network.

![Graphs showing TFCW, TFCW(0.4), TFCW(0.9), and TCP throughput behavior over time.](image)
Smootheness [2/4]

TTFWC with small parameter generates some throughput oscillation comparing to TFWC.
Smoothness [3/4]

TTFWC with large parameter makes larger throughput oscillation than TTFWC with small parameter, but achieves higher throughput.
Smoothness [4/4]

TTFWC with large parameter is as responsive as TCP.

We can observe that TTFWC achieves tunable throughput behavior.
Related work

1. AIMD (Additive Increase and Multiple Decrease) based [1]
   - Reno extension congestion control
   - get lower throughput than TTFWC in congested network.

2. TCP throughput formula-based [2]
   - use TCP throughput equation for congestion window control
   - get more packet loss than TTFWC.

3. Constant Bit Rate (CBR) based and others [3]
   - application layer’s congestion control by UDP
   - not argue about TCP-Friendliness.
   - affect router and middlebox behavior.

Future work

1. Dynamic adaptation for tuning parameters
   - the data rate generated for encoded video varies second by second

2. Limitation of TTFWC design
   - affects protocol friendliness and flows’ throughput
   - When we run only TTFWC flows in the link, they do not achieve fair utilization of the link.