Enhancing TCP for Networks with Guaranteed Bandwidth Services

Yujie Zhu, Oyebamiji Oladeji, Kyu-han Kim and Raghupathy Sivakumar
GNAN Research Group, Georgia Tech
http://www.ece.gatech.edu/research/GNAN/
Introduction

- The Internet is envisioned to be able to provide QoS services in the near future (\texttt{diffserv, intserv})
- Consider applications that can enjoy bandwidth provided by both the \textit{intserv guaranteed service} and the \textit{intserv best effort service}
- Problem Statement: How can a transport layer protocol deliver to such applications the ideal aggregate (reserved + best-effort) bandwidth, while providing TCP’s end-to-end semantics?
- We propose \textbf{GTCP}, an enhanced version of TCP tailored for bandwidth guaranteed environments
TCP over Bandwidth Guaranteed Networks

Bottleneck link 10Mbps, 10ms
f0 – f2: 4Mbps, 2Mbps and 1Mbps reservation
f3 – f5: best effort

Ideal vs. TCP Throughput

Throughput (Mbps)
### TCP Congestion Window Adaptation

- **Slow start:** $cwnd = 1$; $cwnd++$ for every ACK; exit slow start when $cwnd > \text{ssthresh}$
- **Loss indicated by 3 DUPACKs:** $cwnd = cwnd/2$
- **Retransmission Timeout:** $cwnd = 1$, re-enter slow-start
- **Illustration:** reserved bandwidth = 50, $cwnd = 80$

#### TCP Behavior

<table>
<thead>
<tr>
<th>TCP Behavior</th>
<th>Ideal Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original $cwnd$ size = 80</td>
<td>$cwnd_{BE}$</td>
</tr>
<tr>
<td>$cwnd_{BE}/2$ size after 3 DUPACKs</td>
<td>$cwnd_{BE}/2$</td>
</tr>
</tbody>
</table>

TCP's $cwnd$ adaptation is unaware of the reserved bandwidth!
The receipt of an ACK for a packet triggers expansion of congestion window and transmission of a new packet.

However, when there are packet losses, self-clocking is stalled.

Problems exist even when TCP-NewReno is used, as long as the number of packet losses exceed a threshold:

\[ k > \frac{cwnd_{BE}}{2} \]

original cwnd size

outstanding packets after k losses

best effort part of cwnd after cut down

TCP’s self-clocking is unaware of the reserved bandwidth!
Ideal Transport Protocol Design Goals

- **Reserved Bandwidth Awareness**
  Recognize and reliably deliver guaranteed network bandwidth to applications

- **Service Aggregation**
  Achieve aggregation of the best effort and reserved bandwidth available in the network

- **TCP - friendliness**
  Best effort part of the throughput should conform to fairness criteria

- **No Additional Implementation Overhead**
GTCP Design Overview

GTCP is a TCP-friendly transport layer protocol that is reserved bandwidth aware, and delivers to applications the effective aggregate of the reserved and best effort bandwidths

- GTCP uses enhanced congestion window adaptation and self-clocking achieved through tailored mechanisms for *RTT estimation, cwnd calculation, start-up behavior, and congestion control*

- GTCP re-uses TCP’s mechanisms for *flow-control, reliability, sequencing, and connection management*
GTCP Window Adaptation

- $cwnd_G$ estimation
  - $cwnd_G = rate_G \times rtt_{base}$
  - $rtt_{base}$: min round trip time recorded in incoming packets
  - $cwnd = cwnd_G + cwnd_{BE}$

- Start-up behavior
  - Initial $cwnd = 2$
  - $cwnd > ssthresh + cwnd_G$
    - exit slow start

- Diagram:
  - TCP slow start
  - GTCP slow start

- Graph:
  - $cwnd_G + ssthresh$
  - ssthresh

- Time:
  - 1RTT, 2RTT, 3RTT, 4RTT, 5RTT, 6RTT
GTCP Self Clocking

• Transient Congestion
  ➢ Receive 3 DUPACKs:
    ➢ \( cwnd_{update} = cwnd_G + cwnd_{BE}/2 \)
    ➢ \( cwnd \) not reduced immediately
  ➢ For the first \( cwnd_G \) DUPACKs
    ➢ Forced data transmission
    ➢ \( cwnd = cwnd + 1 \)
  ➢ Ignore later \( cwnd_{BE}/2 \) (or \( cwnd_{BE}/2 - k \) when \( k > cwnd_{BE}/2 \)) DUPACKs
  ➢ Transmit new packets for further DUPACKs, if any
  ➢ Full ACK arrival: \( cwnd = cwnd_{update} \)

• Severe congestion
  ➢ Timeout recovery has similar design: at least \( cwnd_G \) packets are transmitted per RTT during timeout
GTCP Self Clocking Illustration

\[ k > \frac{cwnd_{BE}}{2} \]

1. Original cwnd size
   \[ \text{cwnd}_G \quad \text{cwnd}_{BE} \]

2. Outstanding packets after \( k \) losses
   \[ \text{cwnd}_G \quad \text{cwnd}_{BE} \quad \text{cwnd}_G \]

3. Cwnd after the first \( \text{cwndG DUPACKS} \)
   \[ \text{cwnd}_G \quad \text{cwnd}_{BE} \quad \text{cwnd}_G \]

4. Cwnd after the later \( \text{cwnd}_{BE} - k \) \( \text{DUPACKS} \)
   \[ \text{cwnd}_G \quad \text{cwnd}_{BE} \quad \text{cwnd}_G \]

5. New packets sent during fast recovery
   \[ \text{cwnd}_G \]
Simulation Results

- Same topology
- 6 flows (f0 – f5)
  - f0 – f2: 4Mbps, 2Mbps and 1Mbps reservation
  - f3 – f5: best effort

GTCP Throughput

Throughput (Mbps)

- f0
  - Ideal: 4.5
  - TCP: 4.37
  - GTCP: 3.82
- f1
  - Ideal: 2.5
  - TCP: 2.13
  - GTCP: 1.35
- f2
  - Ideal: 1.5
  - TCP: 1.43
  - GTCP: 1.43
Simulation Results (Contd.)

[Graphs showing simulation results for different scenarios]
Other Simulation Results

- Scalability with Link Capacity
- Scalability with Number of Total Flows
- Scalability with Number Flows with reservations
- Scalability with Reserved Bandwidth
- Impact of RTT

GTCP is able to achieve close to ideal throughput in all the above scenarios while maintain TCP-friendliness.
Related Work

- Saha D. Shin et. al.[Transnet’99]:
  - Improve TCP performance with delayed and timed transmission
  - Timer overhead (at granularity of 20ms)

- Lars Wolf et. al.[KiVS’01]:
  - Remove slow start, scale up TCP’s flow control window, rate based transmission
  - No performance comparison with TCP

- Ikjun Yeom et. al.[ICMCS’99]:
  - Inverse rate drop mechanism
  - Require network support
Conclusions & Future work

- Default TCP does not perform well in a bandwidth guaranteed environment
- Reasons for TCP’s non-ideal performance
- GTCP, an transport protocol for achieving ideal throughput in the target environment
- GTCP’s performance verified through simulations
- GTCP implemented in Linux Kernel
- Future work:
  - Non guaranteed QoS: controlled load, diffserv
Questions & Comments?

For more information:
http://www.ece.gatech.edu/research/GNAN/