Implementation and Validation of TCP Options and Congestion Control Algorithms for ns-3

Maurizio Casoni, Carlo Augusto Grazia, Martin Klapez, Natale Patriciello

Department of Engineering Enzo Ferrari
University of Modena and Reggio Emilia

Casteldefels, 14 May 2015
WNS3 ’15
Talk overview

1. Introduction
   - History time
   - State of the Art

2. Background on congestion control

3. Results

4. Conclusions
Section 1

Introduction
TCP Options recap

what
TCP Options: defined in RFC 793, Window Scale and Timestamps presented in RFC 1323

why
To improve performance over large bandwidth*delay product paths and to provide reliable operation over very high-speed paths

where
NS-3 TCP implementation under src/internet
TCP variants recap

**what**
Congestion control algorithms for high-delay links still missing a native implementation for ns-3 (Hybla, Highspeed, ...). Other well-known variant (Bic, Cubic, ...) are in ns-2 but not ported in ns-3.

**why**
The purpose for adding these versions is to update internet module to today’s standard.

**where**
NS-3 TCP implementation under src/internet.
Important TCP options

**Window scale**
Enable the peers to utilize more than 65535 bytes for the sliding window

**Timestamps**
Better RTT estimation, protection for wrap-around sequence number

**SACK**
Selective ACK: explicit knowledge about the segments lost in the network (Breaking news: wns3 2015 poster...)

N. Patriciello (UNIMORE)
Most used TCP congestion control algorithms

**CUBIC/BIC**
Default algorithm in Linux (BIC from 2.6.8 to 2.6.19, CUBIC from 2.6.19 to date)

**Highspeed**
Evolution of NewReno to address its issues in Fast Long-Distance Networks

**Hybla**
Evolution of NewReno to remove its performance dependency from RTT
Section 2

Background on congestion control
What is TCP BIC?

It views the congestion avoidance as a search problem. The congestion window is updated thanks to a binary search, performed from the actual value up to the highest measured value.

Why TCP BIC?

Forerunner of CUBIC, Linux default congestion control algorithm for 2 years.
TCP CUBIC

<table>
<thead>
<tr>
<th>What is TCP CUBIC?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less aggressive and more systematic derivative of BIC, in which the window is a cubic function of time since the last congestion event, with the inflection point set to the window prior to the event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Why TCP CUBIC?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux default congestion control algorithm since 2006.</td>
</tr>
</tbody>
</table>
What is TCP Highspeed?
AIMD parameters changes in function of the current window value. The window will grow faster than standard TCP and also recover from losses more quickly.

Why TCP Highspeed?
Often choosed as a reference benchmark for newer algorithms
What is TCP Hybla?
The key idea is to obtain for long RTT connections the same instantaneous transmission rate of a reference TCP connection with lower RTT

Why TCP Hybla?
Often chose as a reference benchmark for satellite-based network
What is TCP Noordwijk?

A burst-based TCP variant for satellite networks, designed to:

- Optimize transmission of short object
- Guarantee a fair behavior with competing flows
- Operate efficiently over DVB-RCS schemes

Why TCP Noordwijk?

Some interesting ideas for high-delay networks
Section 3

Results
Please note, the max cwnd is determined by the TCP Tx/Rx buffer!
Results: BIC
Results: CUBIC

Cubic cWnd
Cubic ssThresh

Size (KB)
Time (s)

N. Patriciello (UNIMORE) WNS3 '15 14 May 2015 17 / 23
Results: Highspeed

![Graph showing Size (KB) vs. Time (s) for HighSpeed-100ms and NewReno-100ms]

![Graph showing TxData (MB) vs. Time (s) for HighSpeed-100ms and NewReno-100ms]
Results: Hybla
Results: Noordwijk

Throughput (Mbit/s) vs. Time (s)

- Noordwijk flow 0
- Noordwijk flow 1

TxData (MB) vs. Time (s)

- Noordwijk flow 0
- Noordwijk flow 1
- Noordwijk aggregate
- NewReno single flow

N. Patriciello (UNIMORE)
Section 4

Conclusions
Conclusions

Recap
- Presented (and validated) TCP Options infrastructure
- Presented (and validated) TCP congestion control algorithms

Validation Method
Comparison between simulation results with the theoretically expected behaviors

Lesson learned
- Various aspect of TCP still missing testing/validation
- Critical aspect in the TCP layer design
Future Works

Directions to take

- Re-design some part of the internet module to easily extends the TCP (and UDP) testing platform (GSoC ’15 accepted)
- Test TCP for RFC-compliance
- Compare ns-3 and real implementations
Any questions?

natale.patriciello@unimore.it
Why not a comparison with a real stack?

**DCE**
- At the time, DCE was missing the cwnd/ssth tracing capabilities

**Real network**
- Two computer connected with an ethernet cable.
  - side-effects between queue and shaping with tc
  - heavy-optimized TCP (SACK, persistent tables, and so on)
  - really unfair comparison