

L4S simulations with the ns-3 Network Simulator

Tom Henderson
L4S hackathon champion
July 23, 2020

Background

- This slide deck is supplemental to the presentation for the IETF 108 Hackathon on L4S
- Participants: Tom Henderson (champion), Fraida Fund, Bhaskar Kataria Deepak Kumaraswamy, Harsha Sharma, Ashutosh Srivastava
- This covers ns-3 portions only (not Cloudlab)

Motivation

- Low Latency, Low Loss, Scalable Throughput (L4S) is an experimental architecture being worked on by the IETF Transport Area WG
 - <https://www.ietf.org/id/draft-ietf-tsvwg-l4s-arch-06.txt>
- Many early tests have been conducted using small testbeds and Linux kernel implementations
 - <https://github.com/heistp/sce-l4s-bakeoff>
 - <https://github.com/heistp/sce-l4s-ect1>
- ns-3 (discrete-event network simulation) is a complementary tool that can offer larger-scale experiments including wireless core/access link models

L4S models for ns-3

- ns-3 has been used to support the Low Latency DOCSIS^(TM) development, based on the L4S dual queue architecture
- ns-3 models for additional components of L4S have been under development for the past few years, but are not yet in the ns-3 mainline
 - ECN++, AccECN, Dual Queue, ack filtering, FQ/Cobalt
- Endpoint models (TCP Prague, BBRv2, QUIC) supporting L4S are still in early stages

Current ns-3 work on L4S

- Three ns-3 summer students working on L4S-related code
 - Deepak K.: TCP Prague
 - Bhaskar K.: AQMs with L4S support
 - Harsha S.: Flent model, and scripting
- 'l4s-evaluation' extension module contains scripting to support tsvwg scenarios

Background

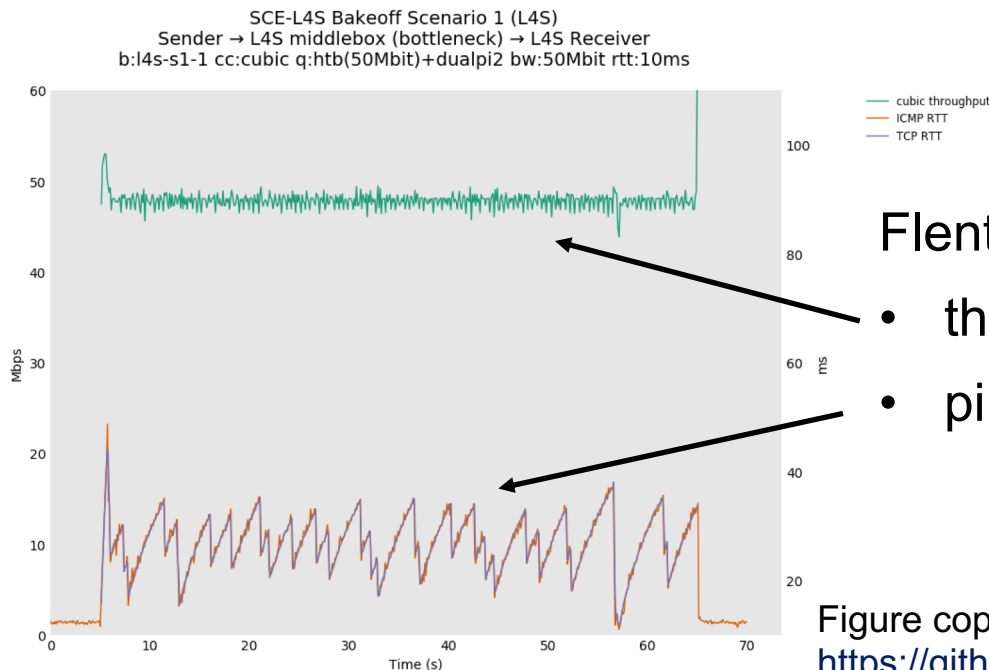
- Pete Heist posted SCE-vs-L4S testbed (Flent-based) data on a GitHub page, prior to two recent tsvwg meetings
- <https://github.com/heistp/sce-l4s-bakeoff>

```
C1 -|                               |- S1
      |- M1 - M2 - M3 - M4 -|
C2 -|                               |- S2
```

- Configuration can support multiple cascading bottlenecks
 - e.g. M4 is a FQ-CoDel bottleneck (50 Mbps)
 - e.g. M3 is a FIFO bottleneck (52.5 Mbps)

Background (cond.)

- Flent (a wrapper around lower-level tools) is used to orchestrate traffic generation and to plot the data
- sample scenario 1, L4S middlebox with cubic, 10ms RTT



Flent typically plots:

- throughput
- ping latency

Figure copied from

<https://github.com/heistp/sce-l4s-bakeoff>

l4s-evaluation module

- Prior to IETF 106 (November), published an extension module to ns-3 to allow it to run simulations similar to Pete Heist's experiments
 - draft-compliant dual queue coupled AQM model
 - TCP DCTCP as a surrogate for TCP Prague (using a TCP Reno startup algorithm)
 - simple ns-3 TCP bulk send applications
 - custom plotting scripts

ns-3 simulations from I4s-evaluation

- ns-3 data traces gathered

```
// tsvwg-scenarios.cc responds to issues from the IETF tsvwg tracker
//
// ----> downstream (data transfer client to server)
// <--- upstream (return acks and ICMP echo response)
//
//      (1)  (2)  (3)  (4)
// client ---| M1 |----| M2 |----| M3 |--- server
//           |   |   |   |   |
// node 0     1     2     3     4 (ns-3 node IDs)
//
// - The box M1 is notionally access network headend such as a CMTS or DSLAM
// - The box M2 is notionally access network CPE device such as a cable or DSL
//   modem.
// - The box M3 is notionally a home router (HR) running cake or FQ-CoDel
//
```

- 1) ICMP observed RTT latency (every 100 ms)
- 2) TCP RTT samples (each RTT estimate)
- 3) TCP throughput (averaged every 200 ms)

ns-3 simulations (cont)

- ns-3 data traces gathered (cont)

```
// tsvwg-scenarios.cc responds to issues from the IETF tsvwg tracker
//
// ----> downstream (data transfer client to server)
// <--- upstream (return acks and ICMP echo response)
//
//      (1)  (2)  (3)  (4)
// client ---| M1 |---| M2 |---| M3 |--- server
//           |   |   |   |
// node 0     1     2     3     4 (ns-3 node IDs)
//
// - The box M1 is notionally access network headend such as a CMTS or DSLAM
// - The box M2 is notionally access network CPE device such as a cable or DSL
//   modem.
// - The box M3 is notionally a home router (HR) running cake or FQ-CoDel
//
```

- 4) TCP cwnd trace at client (sender)
- 5) M3 ECN mark frequency (every 100ms)
- 6) M2 drop frequency (every 100ms)

TCP Prague model

- TCP Prague is based on DCTCP with the following extensions
 - new flow startup for bandwidth probing (paced chirping)
 - RTT independence (bias) for fairness with legacy flows
 - fallback to TCP Reno upon loss detection
 - AccECN support (AccECN TCP option)
 - RFC-3168 queue detection heuristics

Hackathon
work

Low-latency support for AQMs

- IETF dual queue implemented since 2018
- FQ/CoDel did not yet have L4S support
 - step threshold for L4S ECT(1) traffic added by Bhaskar and integrated this week
- Next steps include FQ/Cobalt with BLUE enhancement and L4S
- Possible other enhancements to L4S threshold such as a ramp

Hackathon
work

Flent support

- Flent coordinates data transfers between clients and servers, and collects throughput and latency measurements in a JSON-formatted file
 - Flent then passes this file through plotting scripts
- ns-3 support (in progress by Harsha S.) will support a Flent-like API and generate JSON data files, for plotting by Flent

L4S testing on cloud testbeds

- Ashutosh Srivastava and Fraida Fund (NYU Wireless) has experimented with TCP Prague and BBR implementations on CloudLab
 - INFOCOM 2020 publication
- As part of hackathon, Ashutosh and Fraida reproduced some of Pete Heist's results, and worked towards automating the CloudLab profiles to make it easy for new users

Integrated tree for IETF hackathon

- ‘hackathon/master’ branch of <https://gitlab.com/tomhenderson/ns-3-dev>
- This is the ns-3-dev mainline plus
 - TCP Linux-like pacing
 - Dual Queue model
 - current TCP Prague model
 - FQ/CoDel with L4S support
- Also relies on a supplemental extension module for L4S evaluation
 - ‘hackathon/master’ branch of <https://gitlab.com/tomhend/modules/l4s-evaluation.git>

Installation quick start

- Prerequisites for ns-3:
 - Python 3 and a C++ compiler (g++ or clang++)
- Prerequisites for L4S plotting:
 - Python numpy and matplotlib, pdfjam
- Linux, BSD, or macOS
 - Windows Visual Studio not supported

Installation quick start (cont.)

- download the following fork of ns-3:

```
git clone https://gitlab.com/tomhenderson/ns-3-dev.git
```

- check out the 'hackathon/master' branch

```
cd ns-3-dev
```

```
git checkout -b hackathon/master  
origin/hackathon/master
```

Installation quick start (cont.)

- check out the l4s-evaluation extension (also using the hackathon/master branch) into the 'contrib' directory:

```
cd contrib
```

```
git clone https://gitlab.com/tomhend/modules/l4s-evaluation.git
```

```
cd l4s-evaluation
```

```
git checkout -b hackathon/master  
origin/hackathon/master
```

- change back to top-level ns-3 directory

```
cd ..
```

Installation quick start (cont.)

- Build ns-3

```
./waf configure --enable-examples --enable-tests -d  
optimized
```

- Run all unit tests to check if all pass

```
./test.py
```

Run an experiment

- Run a scripted experiment; a single bottleneck queue, TCP Prague flow

```
cd contrib/l4s-evaluation/experiments/tsvwg-issue-17-one-flow
```

- edit the script 'run-single.sh' (single simulation execution) using your favorite editor

```
vi run-single.sh
```

Editing the configuration

```
# Run a single instance of tsvwg issue 17 scenario 6, one TCP flow
#
# Basic configuration to replicate scenarios from Pete Heist's results:
# scenario 2 (issue 16):  controlScenario=1, m3QueueType=fq, ...
# scenario 3 (issue 16):  controlScenario=1, m3QueueType=codel, ...
# scenario 5 (issue 17):  controlScenario=0, m3QueueType=fq, link3rate=52.5Mbps,
#                          link5RateRatio=0.9524, ...
# scenario 6 (issue 17):  controlScenario=0, m3QueueType=fq, ...

#####
# scenario details
scenario_id=one-flow
link3rate=50Mbps
link5rateRatio=0.95
rtt=80ms
controlScenario=1
enablePcap=0
firstTcpType=cubic
m3QueueType=fq
RngRun=1
#####
```

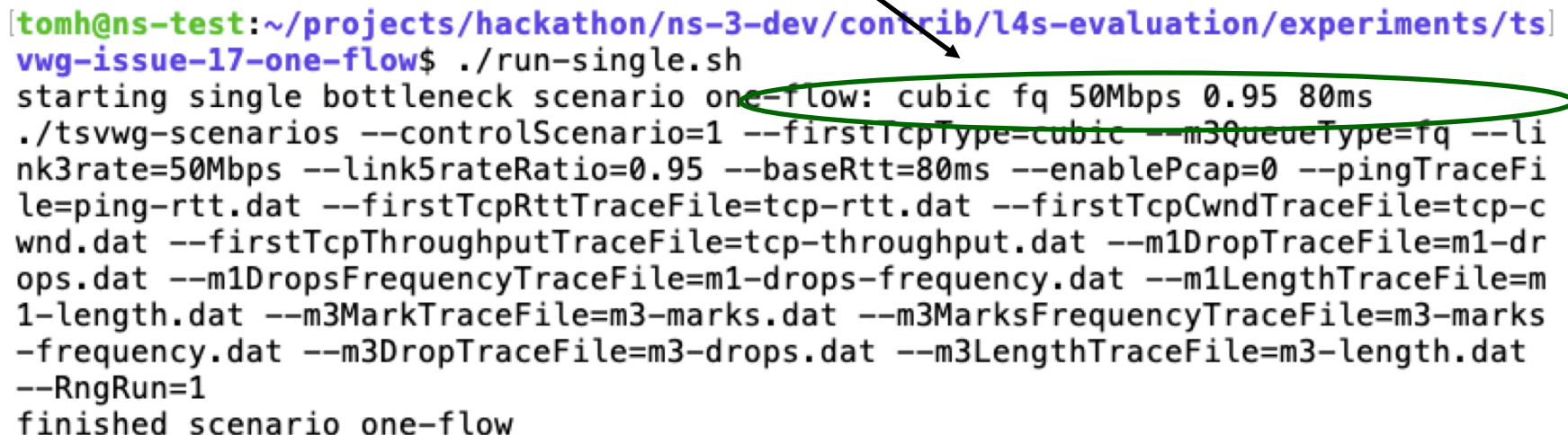
Choices are 'prague', 'cubic', 'reno'

Choices are 'fq', 'dualq', 'codel'

Run the script

- After editing parameters, call 'run-single.sh'
- The script echoes the configuration passed to the ns-3 'tsvwg-scenarios' program

Single cubic flow, fq AQM, ~50 Mbps bottleneck, 80ms base RTT



```
[tomh@ns-test:~/projects/hackathon/ns-3-dev/contrib/l4s-evaluation/experiments/tsvwg-issue-17-one-flow$ ./run-single.sh
starting single bottleneck scenario one-flow: cubic fq 50Mbps 0.95 80ms
./tsvwg-scenarios --controlScenario=1 --firstTcpType=cubic --m3QueueType=fq --link3rate=50Mbps --link5rateRatio=0.95 --baseRtt=80ms --enablePcap=0 --pingTraceFile=ping-rtt.dat --firstTcpRttTraceFile=tcp-rtt.dat --firstTcpCwndTraceFile=tcp-cwnd.dat --firstTcpThroughputTraceFile=tcp-throughput.dat --m1DropTraceFile=m1-drops.dat --m1DropsFrequencyTraceFile=m1-drops-frequency.dat --m1LengthTraceFile=m1-length.dat --m3MarkTraceFile=m3-marks.dat --m3MarksFrequencyTraceFile=m3-marks-frequency.dat --m3DropTraceFile=m3-drops.dat --m3LengthTraceFile=m3-length.dat --RngRun=1
finished scenario one-flow
```

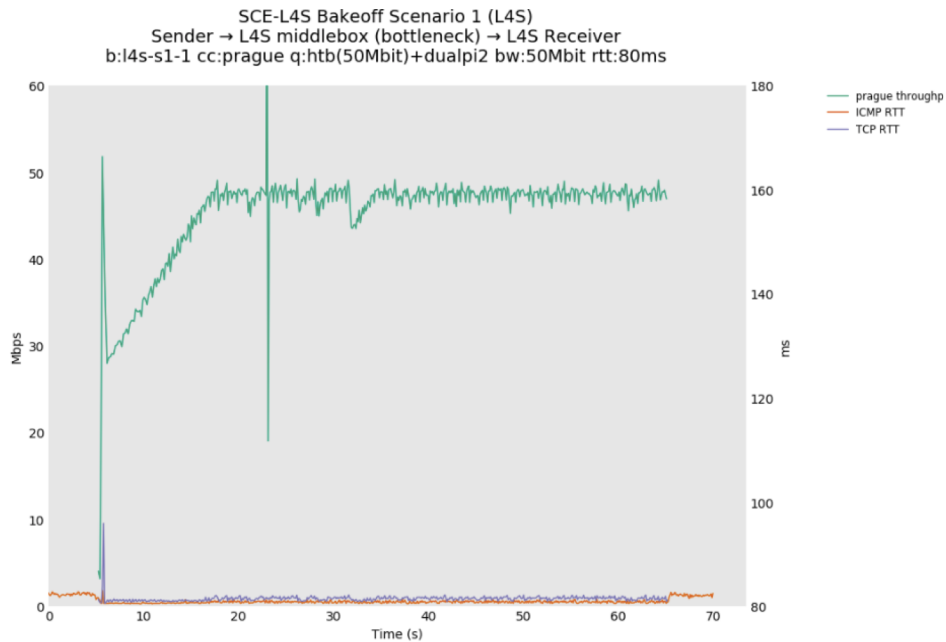
Results

- All data will be in a timestamped 'results' directory, unique for each simulation run
- A script generates a multi-plot with various time-series plotted

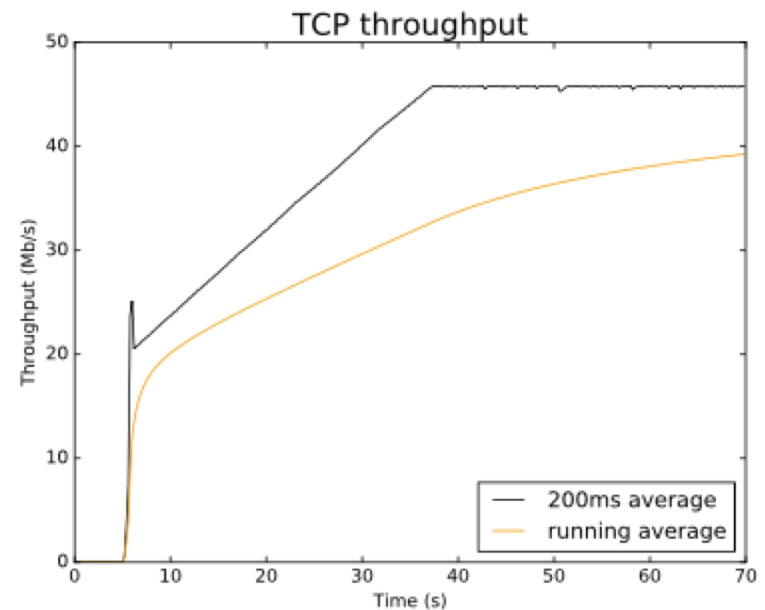
```
-flow/results/one-flow-20200722-165532$ ls
log.txt
m1-drops.dat
m1-drops-frequency.dat
m1-drops-frequency.pdf
m1-drops.pdf
m1-length.dat
m1-length.pdf
m1-length-zoom.pdf
m3-depth.pdf
m3-drops.dat
m3-length.dat
m3-marks.dat
m3-marks-frequency.dat
m3-marks-frequency.pdf
one-flow-cubic-fq-1-50Mbps-0.95-80ms.pdf
ping-rtt.dat
ping-rtt.pdf
plot-m1-drops-frequency.py
plot-m1-drops.py
plot-m1-length.py
plot-m3-length.py
plot-m3-marks-frequency.py
plot-ping-rtt.py
plot.sh
plot-tcp-cwnd.py
plot-tcp-rtt.py
plot-tcp-throughput.py
run-single.sh
tcp-cwnd.dat
tcp-cwnd.pdf
tcp-rtt.dat
tcp-rtt.pdf
tcp-throughput.dat
tcp-throughput.pdf
tsvwg-scenarios
tsvwg-scenarios.cc
tsvwg-scenarios-second-tcp-cwnd.dat
tsvwg-scenarios-second-tcp-rtt.dat
tsvwg-scenarios-second-tcp-throughput.dat
version.txt
```

Basic single-flow Prague comparison

Linux



ns-3



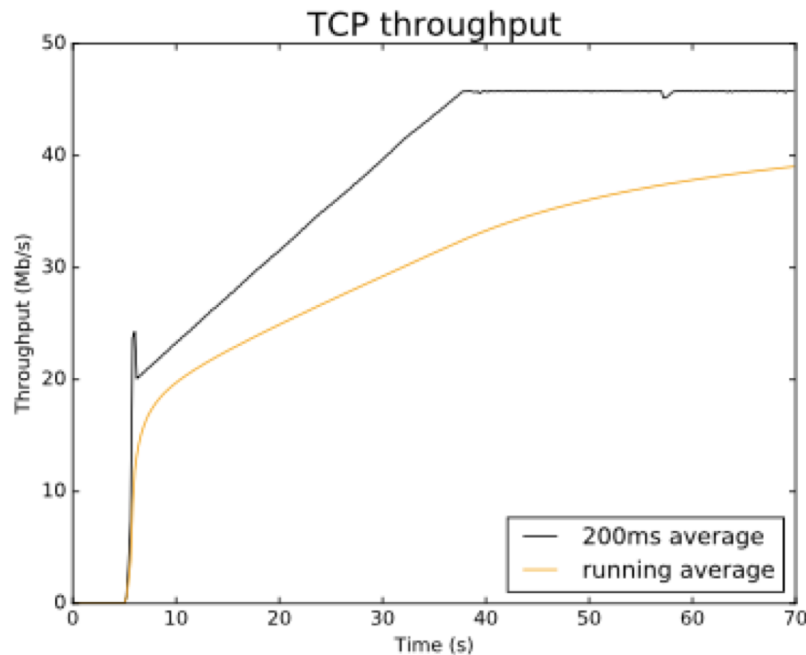
https://www.heistp.net/downloads/sce-l4s-bakeoff/bakeoff-2019-11-11T090559-r2/l4s-s1-1/batch-l4s-s1-1-prague-50Mbit-80ms_fixed.png

Differences to explore in slow-start termination, slope of recovery

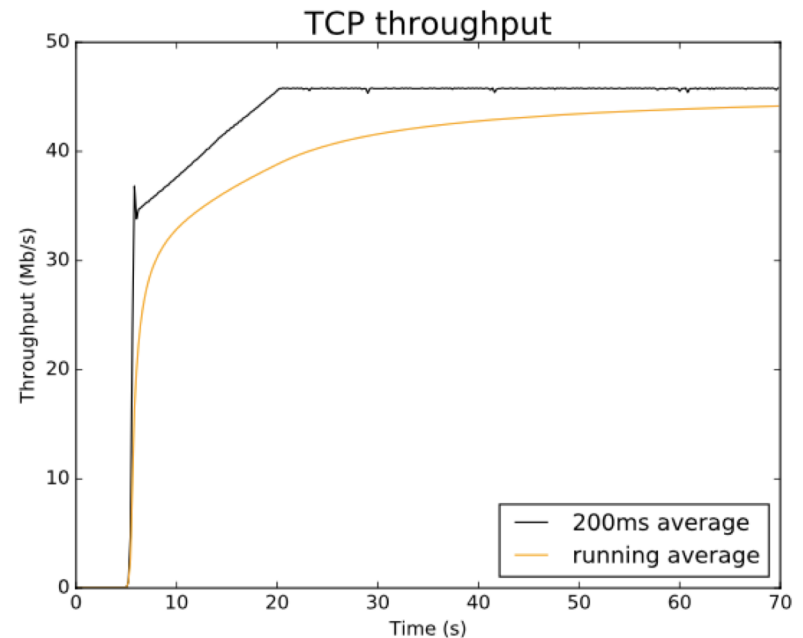
Slow start differences

- Prague can be run with Linux-like pacing, or with a proposed 'Paced Chirping' startup
 - configuration options in the 'one-flow' run-single.sh

Linux pacing



Paced Chirping



Next steps

- This version of ns-3 TCP Cubic does not support ECN
 - Cubic needs to be finalized for the ns-3 mainline
- Much more work needed on TCP Prague model
- Move away from Bash scripts to use of SEM (simulation execution manager), possibly make the previous slides into a runnable Jupyter notebook
- Add Flent plotting for closer alignment to testbed results