Progress Report: Make NDN Congestion Control work in ndnSIM

6th NDN Hackathon

Klaus Schneider, Ashiqur Rahman, Chavoosh Ghasemi

May 13, 2018

The University of Arizona

Need

Congestion Control = Crucial part of **App Performance**

Motivation and Contribution to NDN:

- The current NFD congestion detection doesn't work in ndnSIM, since ndnSIM doesn't use regular TCP, UDP, or Unix faces.
- This project aims to fix this and evaluate the proper function of the implementation.
- Tasks:
  - Implement a ns-3 Queue based on CoDel that detects congestion and inserts congestion marks.
  - Map these congestion marks onto NDNLP/ns-3 packets.
  - Implement a ndnSIM consumer application that reacts to the congestion marks.
- Additional Tasks (if time):
  - Start submitting the code to Gerrit.
  - Implement a TCP-Cubic like consumer app and compare

Ethernet Queue

How to access low-level queues?

- libnl library
- OS-dependent (Linux)
- Hard to use (very low-level API)

⇒ Couldn't get it to work.

UDP & TCP Tunnels

How to access socket queue?

- ioctl commands: TIOCOUTQ, SIOCOUTQNSD
- Portable (Mac and Linux)
- Retrieve queue backlog size.
- Threshold: 50KB (out of 200KB buffer limit)

UDP & TCP Tunnels

How to access socket queue?

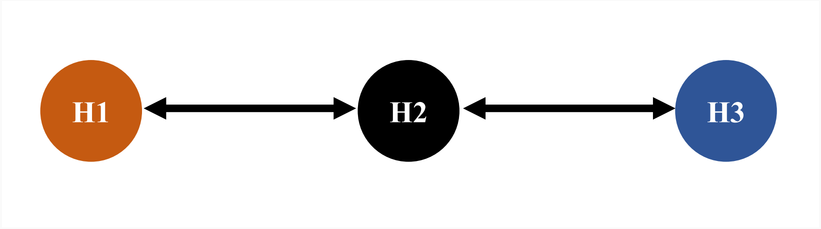
- ioctl commands: TIOCOUTQ, SIOCOUTQNSD
- Portable (Mac and Linux)
- Retrieve queue backlog size.
- Threshold: 50KB (out of 200KB buffer limit)

TCP:

- Also look at std::queue<Block> m_sendQueue
- Threshold: 10 pkts (out of infinite)

Experimental environment

Local topology:



- UDP Tunnels & TCP Tunnels
- Ethernet & WiFi

Results (WiFi):

UDP (both links):

Scenario	RTT	Goodput	Retx
Without CC:	60ms	28.7 Mbps	250 Retx
With CC:	6-9ms	26.0 Mbps	0 Retx

Results (WiFi):

UDP (both links):

Scenario	RTT	Goodput	Retx
Without CC:	60ms	28.7 Mbps	250 Retx
With CC:	6-9ms	26.0 Mbps	0 Retx

TCP (both links):

Scenario	RTT	Goodput	Retx
Without CC:	115ms	37 Mbps	320 Retx
With CC:	7ms	29 Mbps	0 Retx

Results (WiFi):

UDP (both links):

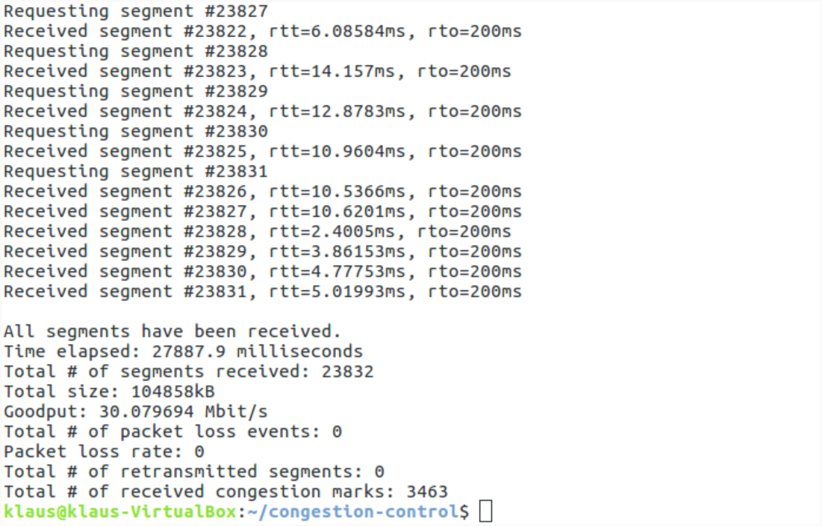
Scenario	RTT	Goodput	Retx
Without CC:	60ms	28.7 Mbps	250 Retx
With CC:	6-9ms	26.0 Mbps	0 Retx

TCP (both links):

Scenario	RTT	Goodput	Retx
Without CC:	115ms	37 Mbps	320 Retx
With CC:	7ms	29 Mbps	0 Retx

Ethernet: Cong. signals can even give you a **higher goodput!**

Results (WiFi):



```
Requesting segment #23827
Received segment #23822, rtt=6.08584ms, rto=200ms
Requesting segment #23828
Received segment #23823, rtt=14.157ms, rto=200ms
Requesting segment #23829
Received segment #23824, rtt=12.8783ms, rto=200ms
Requesting segment #23830
Received segment #23825, rtt=10.9604ms, rto=200ms
Requesting segment #23831
Received segment #23826, rtt=10.5366ms, rto=200ms
Received segment #23827, rtt=10.6201ms, rto=200ms
Received segment #23828, rtt=2.4005ms, rto=200ms
Received segment #23829, rtt=3.86153ms, rto=200ms
Received segment #23830, rtt=4.77753ms, rto=200ms
Received segment #23831, rtt=5.01993ms, rto=200ms

All segments have been received.
Time elapsed: 27887.9 milliseconds
Total # of segments received: 23832
Total size: 104858kB
Goodput: 30.079694 Mbit/s
Total # of packet loss events: 0
Packet loss rate: 0
Total # of retransmitted segments: 0
Total # of received congestion marks: 3463
klaus@klaus-VirtualBox:~/congestion-control$
```

Takeaway:

`https://github.com/5th-ndn-hackathon/congestion-control`

1. Congestion detection by socket queue works!
2. Even works on WiFi (much lower layer).
3. Use UDP Tunnels + congestion signaling everywhere.

Takeaway:

https://github.com/5th-ndn-hackathon/congestion-control

1. Congestion detection by socket queue works!
2. Even works on WiFi (much lower layer).
3. Use UDP Tunnels + congestion signaling everywhere.

Future work

1. Implement in NFD (run on testbed)
2. Test “hidden congestion” inside IP underlay (tunnel)

The End

Any Questions?

Klaus Schneider, Eric Newberry, Chavoosh Ghasemi