Improving Multipath TCP Congestion Control

Cyrus Tabatabai-Yazdi, Zongheng Ma, Kimberly Chou,
Akshay Shetty, Akshay Raman, Alan Tang,
Sean Xiaowen Zhang

Background

Multipath TCP

- Networks are multi-path
  - Multiple radios for mobile devices
  - Multiple parallel paths in data centers through ECMP
- Utilize those paths for performance improvements
- In lieu of these problems, MPTCP was proposed and developed as variant of TCP that can transmit among several paths simultaneously

Benefits of Multipath TCP

- Reliability
- Load balancing across multiple interfaces

Design Goal: MPTCP

- Application layer directly uses the API the same way as for regular TCP
  - Maintain regular TCP service model and API
  - Look like regular TCP as it traverses network
- Work in all scenarios and can fall back to TCP if needed
- Perform at least as well as regular TCP without starving TCP
  - Co-exist gracefully with other TCP flows

Design Goal: MPTCP

- Connection setup
- Transmission and acknowledgements
- Congestion control
- Flow control
- Connection teardown handshake and state machine

Design Goal: MPTCP

- Connection setup
- Transmission and acknowledgements
- **Congestion control**
- Flow control
- Connection teardown handshake and state machine

We focused on congestion control in this project

Design Goal: Congestion Control

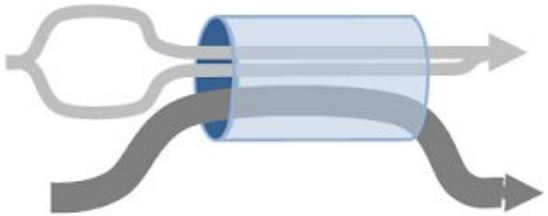
- Fair to regular TCP connections
- Use efficient paths
- Quick adaptation
- Traffic is moved to less congested subflows.

Existing Algorithms: Regular TCP

- Each ACK, increase the congestion window *w* by *1/w*
- Each loss, decrease *w* by *w/2*

Existing Algorithms: Regular TCP

- It does not achieve fairness when used for MPTCP!



Existing Algorithms: MPTCP

- Each ACK on subflow r, for each subset S of R that includes path r, compute:

$$\alpha = \frac{\max_{s \in S} w_s / \mathrm{RTT}_s^2}{\left(\sum_{s \in S} w_s / \mathrm{RTT}_s\right)^2},$$

- Then find the minimum over all such S, and increase w_r by that much. (The complexity of finding the minimum is linear in the number of paths.)
- Each loss on subflow r, decrease the window w_r by $w_r/2$.

Existing Algorithms: Equally Weighted TCP

- For each ACK on path r, increase window w_r by a/w_r.
- For each loss on path r, decrease window wr by $w_r/2$.

W_r: Window size on path r

a: 1/sqrt(n), where n is number of paths

Existing Algorithms: EWTCP

- For each ACK on path r, increase window w_r by a/w_r.
- For each loss on path r, decrease window wr by $w_r/2$.

Achieves fairness, but it is not efficient

Existing Algorithms: Coupled

- For each ACK on path r, increase window w_r by $1/w_{total}$.
- For each loss on path r, decrease window wr by $w_{total}/2$.

W_{total}: Total window size across all subflows

Existing Algorithms: Semi-coupled

- For each ACK on path r, increase window w_r by a/w_{total}.
- For each loss on path r, decrease window wr by $w_r/2$.

Existing Algorithms: Linked Increase

In Congestion Avoidance Phase:

- For each ACK on path i, increase congestion window for i by

$$\frac{\alpha \cdot bytes\_acked \cdot mss_i}{tot\_cwnd}$$

where $$\alpha = tot\_cwnd \frac{\max_i \frac{cwnd_i \cdot mss_i^2}{rtt_i^2}}{\left(\sum_i \frac{cwnd_i \cdot mss_i}{rtt_i}\right)^2}$$

Existing Algorithm: RTT Compensator

- For each ACK on subflow r, increase window w_r by:
  - $\min(a/w_{total}, 1/w_r)$
- For each loss on subflow r, decrease window w_r by $w_r/2$

Motivation

Problem Statement

- MPTCP could lead to performance gains through the use of multiple paths
- *But..*

Problem Statement

- How do MPTCP congestion control algorithms perform in different situations? Are they a one size fits all?
- Can we improve them?

Our Project

- We evaluated the existing MPTCP congestion control algorithms in different topologies using ns-3 simulator
- We designed and evaluated a new congestion control algorithm, taking into account factors like RTT and number of consecutive acknowledgements received to dynamically adjust the rate at which the congestion window changes per subflow
- Looked at low rate mobile ad hoc networks as our main usage scenario

Methods

Current Scenario

1. We know that current MPTCP algorithms are aggressive on fairness.
2. What about situations where we have a high speed paths and is currently underutilized ?
3. Current solutions are very slow on increasing cwnd during times of no/least congestion.
4. We thought about a solution that helps rapidly increase the cwnd upon underutilized paths.

Delay based mechanism for faster recovery

1. TCP Compound is helpful introduces total congestion window as the sum of cwnd (window based on packet drops) and dwnd (a delay based window that depends on the moving RTT average) .
2. We have incorporated the dwnd mechanism for our solution .

Total cwnd of a subflow = cwnd + dwnd

Dwnd = alpha * bestRTT/lastMeasuredRtt

$$\text{Alpha} = \text{cwnd\_total} * \frac{\text{MAX}(\text{cwnd\_i}/\text{rtt\_i}^2)}{(\text{SUM}(\text{cwnd\_i}/\text{rtt\_i}))^2}$$

Our Solution

- On drop of packets , the procedure remains the same as other mptcp congestion algorithms, mostly decrease by two. (Wr/2)
- However upon receiving ACKs continuously , we include a component Dwnd [Delay window]

```
if ( ack_count > 5 ) {
Dwnd =  alpha * bestRTT/lastMeasuredRtt
}
```

- bestRTT - is the best RTT value from the subflow path.
- lastMeasuredRtt is the last measured RTT for the subflow
- Based on the improvement of last returned RTT , we add a larger amount to cwnd.

Our Solution

- Best case scenerio :

  lastMeasuredRtt = bestRTT ;

  Dwnd = alpha and

  cwnd = cwnd + alpha = 2* alpha

We increase the cwnd by twice the amount then the traditional RTT Compensator.

Fairness issue on subflow

- Although we want to rapidly fill up the underutilized path , we also want to make sure that we are not adding to existing bottleneck
- Our algorithm is based on the following assumptions.
  - LastMeasuredRTT provides better overview about the congested path. The smaller the value, the less congestion on the path.
  - We use dwnd only when the number of ACKs received is more than 5 (we are still fine tuning what parameter would be better for different topologies)
  - Set ACK count to 0 when packets are dropped. In that case, the TCP will go into slow start and be based only on earlier cwnd and alpha.
  - Only after 5 ACKS and better lastMeasuredRTT , we rapidly increase the cwnd.

Implementation Overview

- Run simulation on ns-3 simulator
- Used mptcp model for ns-3
- Designed and built topologies in ns-3
- Tested on existing algorithms and our algorithm

ns-3

- Open-source network simulator
- Written in C++
- Supports various types of networks
  - IP or non-IP
  - Wi-Fi
  - LTE

MPTCP model in ns-3

- Developed by previous researchers
- Split network traffic to multiple subflows

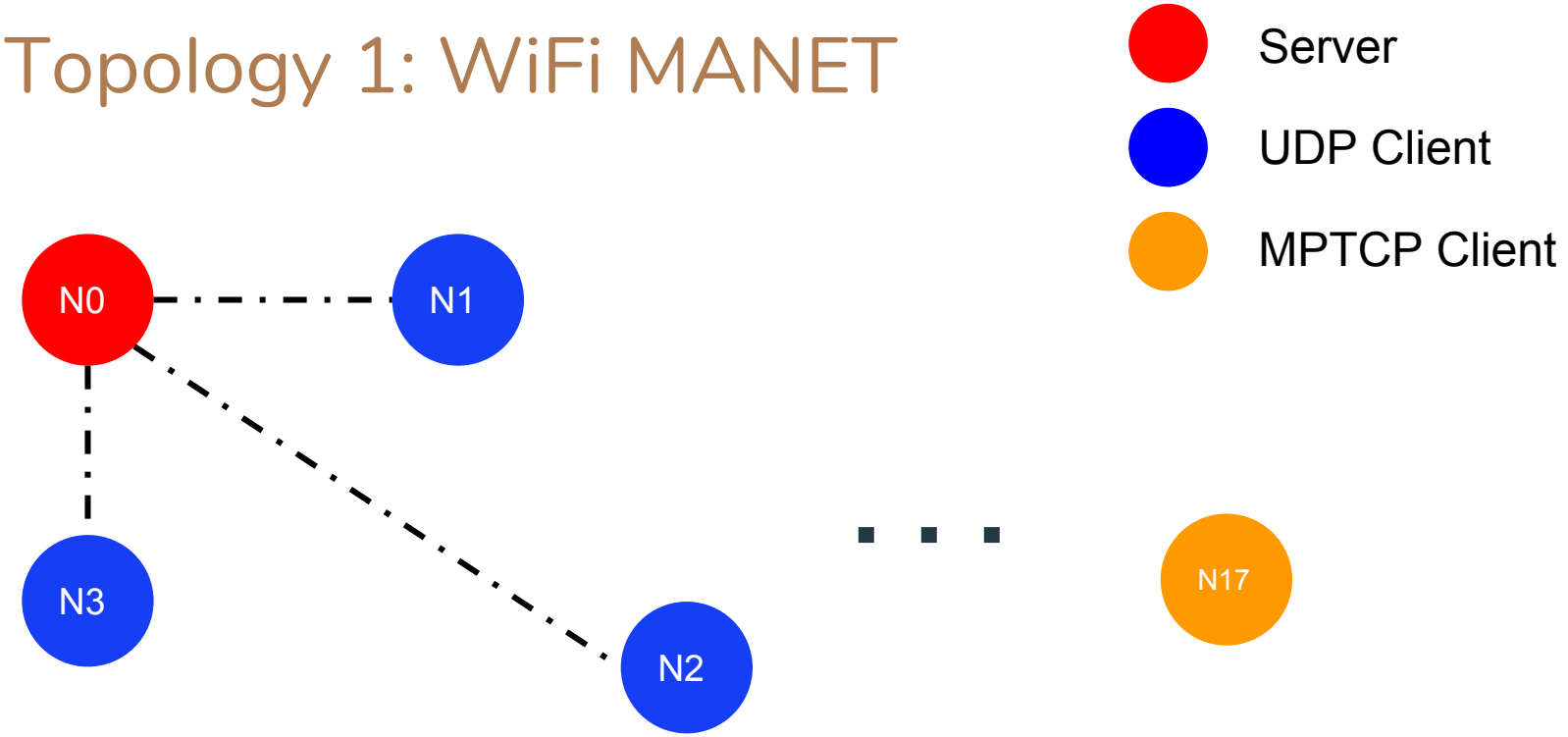
Topology

- We implemented 2 topologies and 2 additional topology is in progress
  - WiFi MANET
  - CSMA
  - CSMA + WiFi (In progress)
  - Triangular (In progress)

Topology 1: WiFi MANET

- 18 Nodes (Maximum number supported in ns-3 mobility model)
- Connected by WiFi
- Mobility model represented by random walk

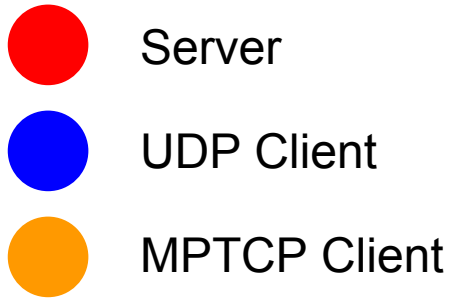
Topology 1: WiFi MANET

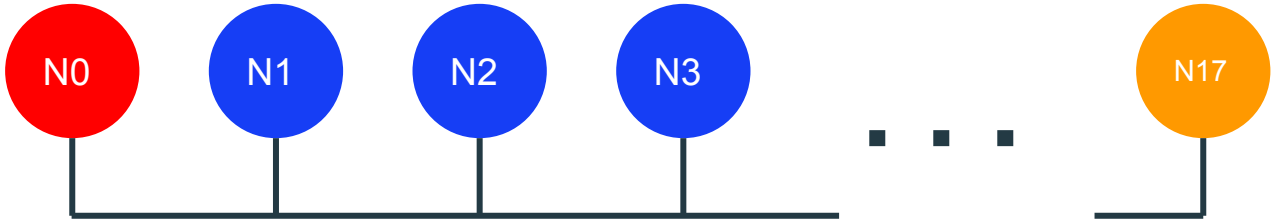


Topology 2: CSMA

- 18 Nodes
- Connected by Wire
- No Mobility
- Simulate Ethernet

Topology 2: CSMA

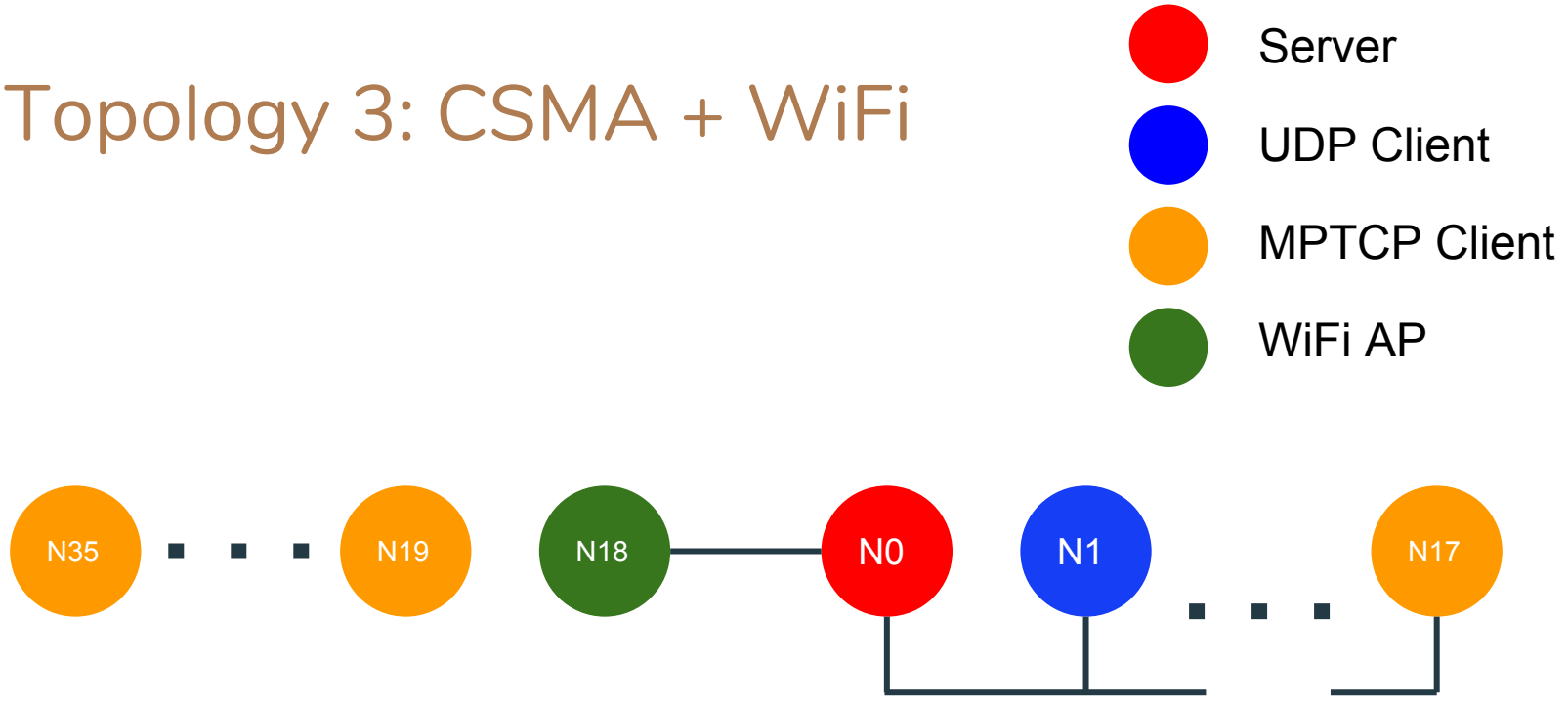




Topology 3: CSMA + WiFi (In Progress)

- 36 Nodes
- Combination of previous two topologies
- WiFi Access Point is connected with CSMA network via a point-to-point link

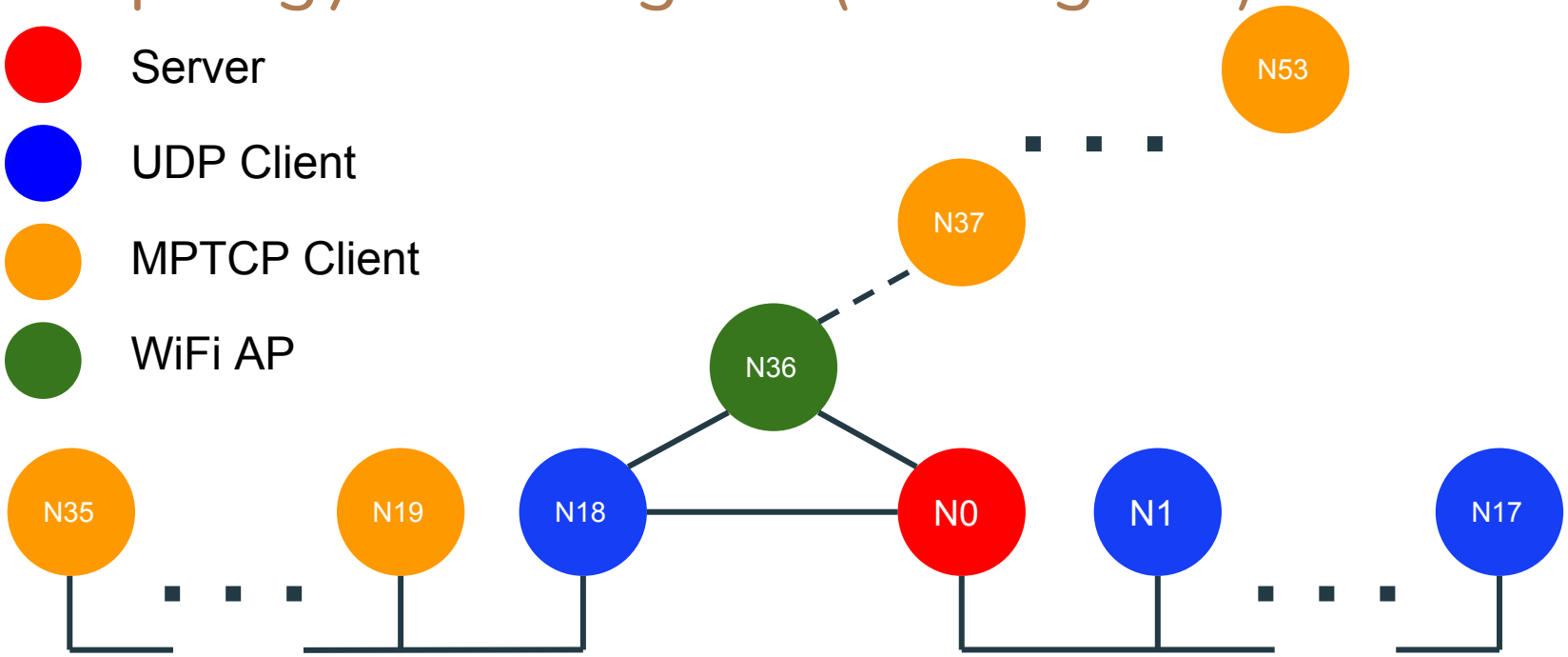
Topology 3: CSMA + WiFi



Topology 4: Triangular (In Progress)

- 54 Nodes
- Two sets of cmsa nodes
- One set of Wi-Fi node
- Multiple physical paths instead of subflows

Topology 4: Triangular (In Progress)



Challenges

- Difficulty in developing in ns-3
  - Multiple subsystems work together
- Fairness Testing with a Regular TCP connection
  - Bug in ns-3 BulkSendApp
    - Not allowing tcp bulk sender to co-exist with MPTCP bulk sender
    - Unable to have both regular tcp node and MPTCP node in topology
- LTE Complications

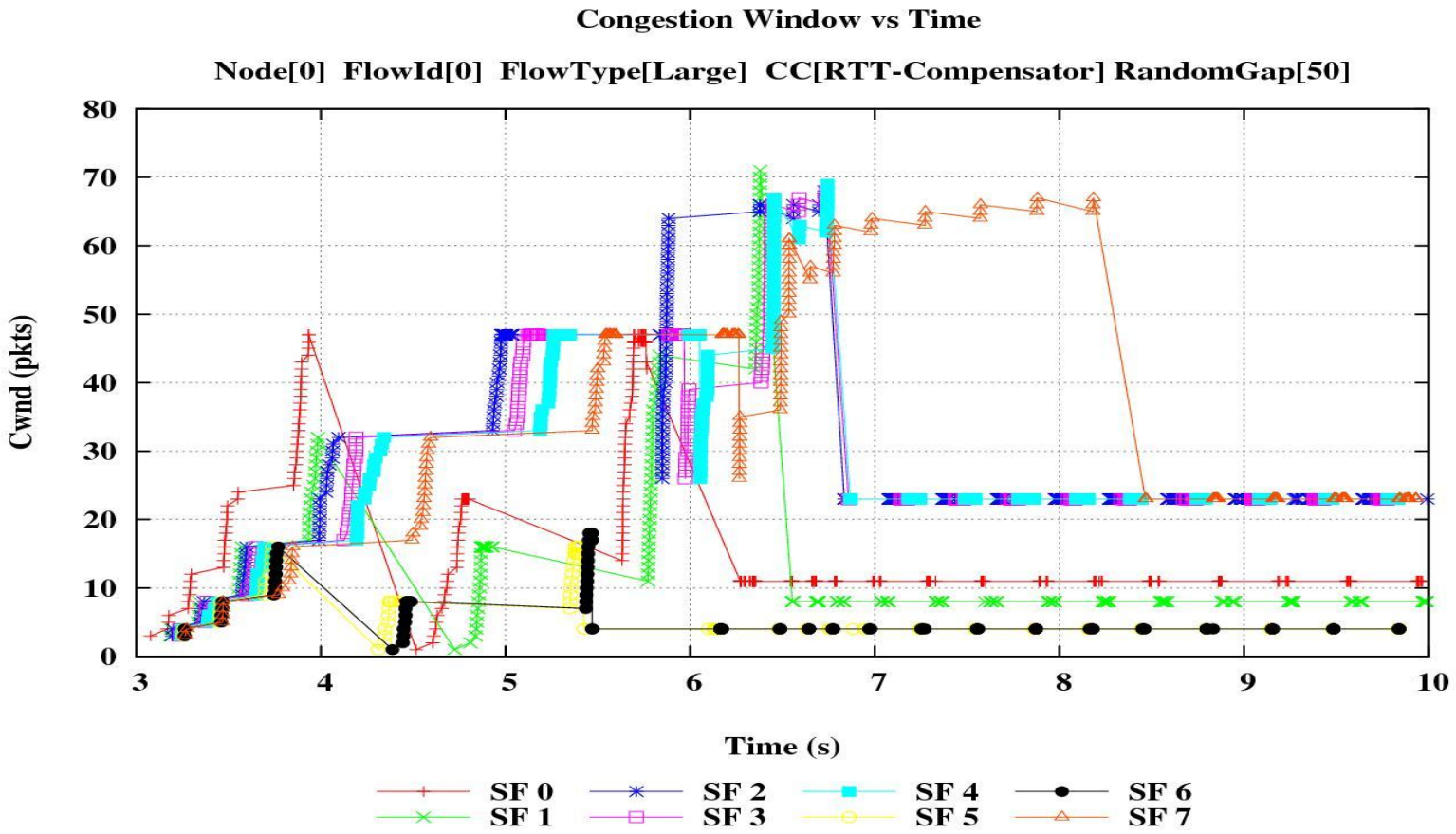
Demo

Results and Evaluations

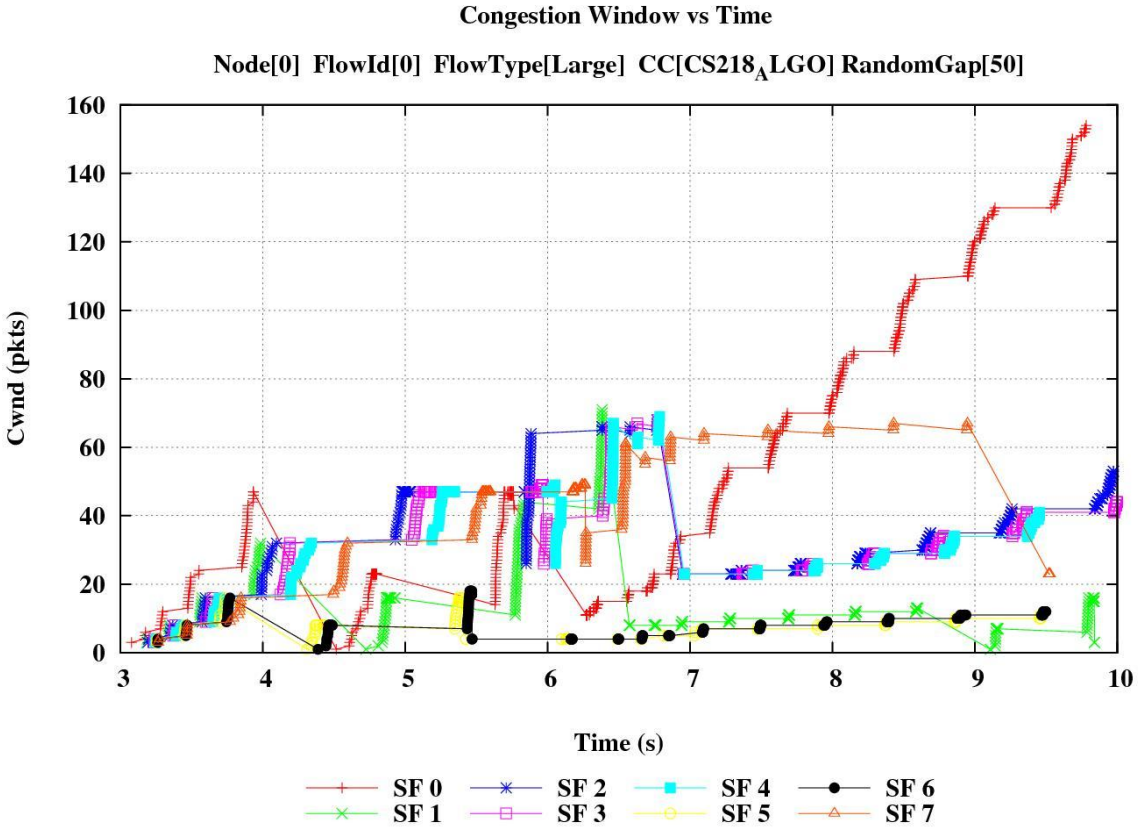
Evaluation Metrics

- Look at the time varying RTT and CWND to gauge effectiveness of algorithm
- In particular, look at RTT since it is a better measure of whether subflows are being effectively utilized.
- Graphs shown use the topology #1: WiFi MANET

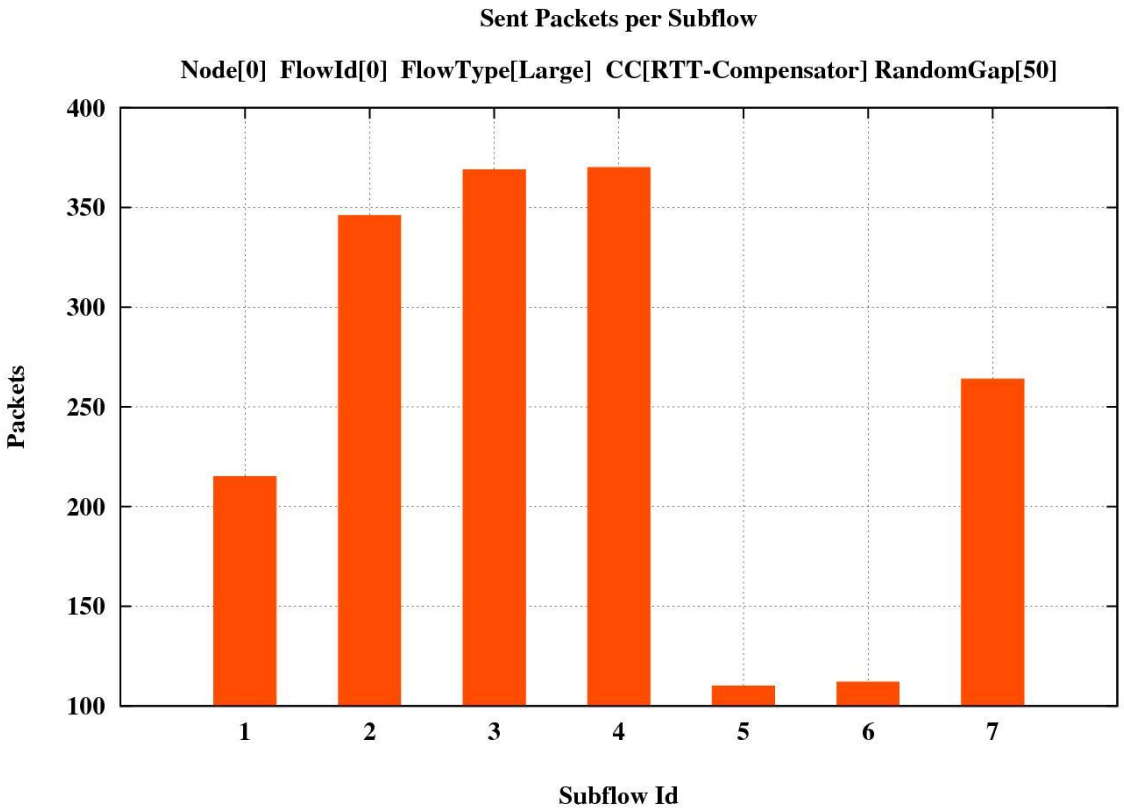
RTT Compensator



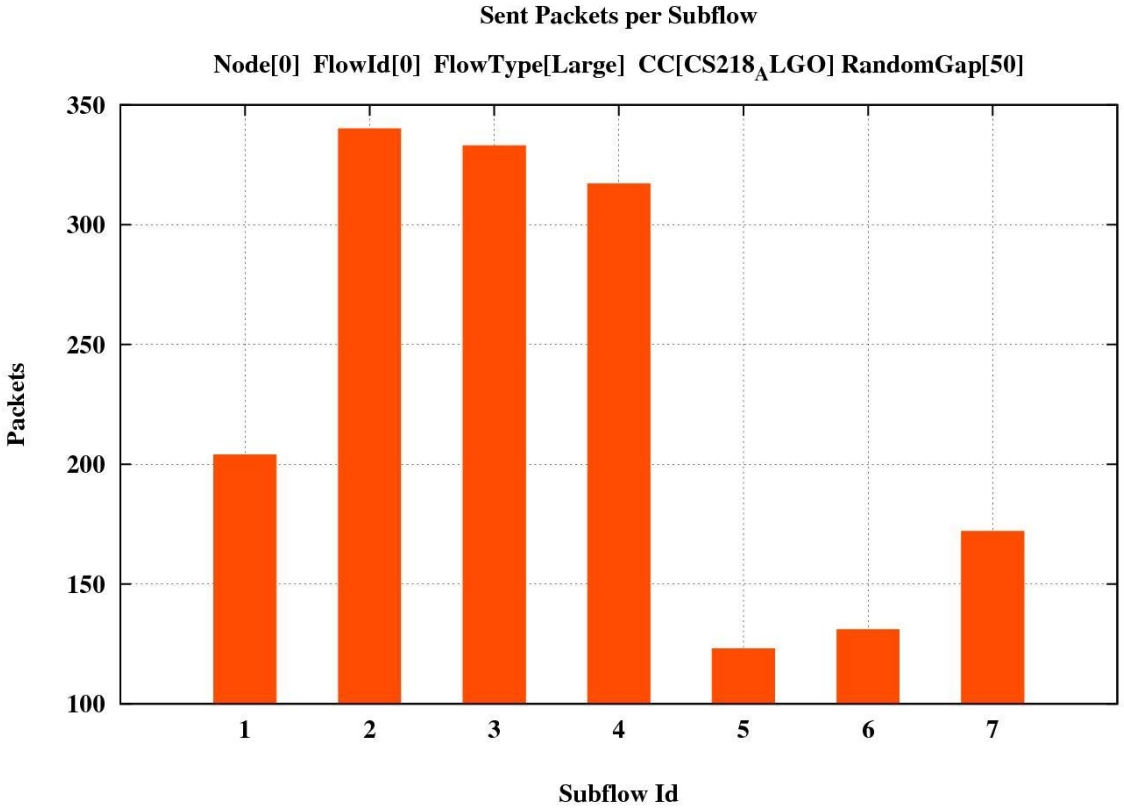
Delay + ACK-count based increase for cwnd



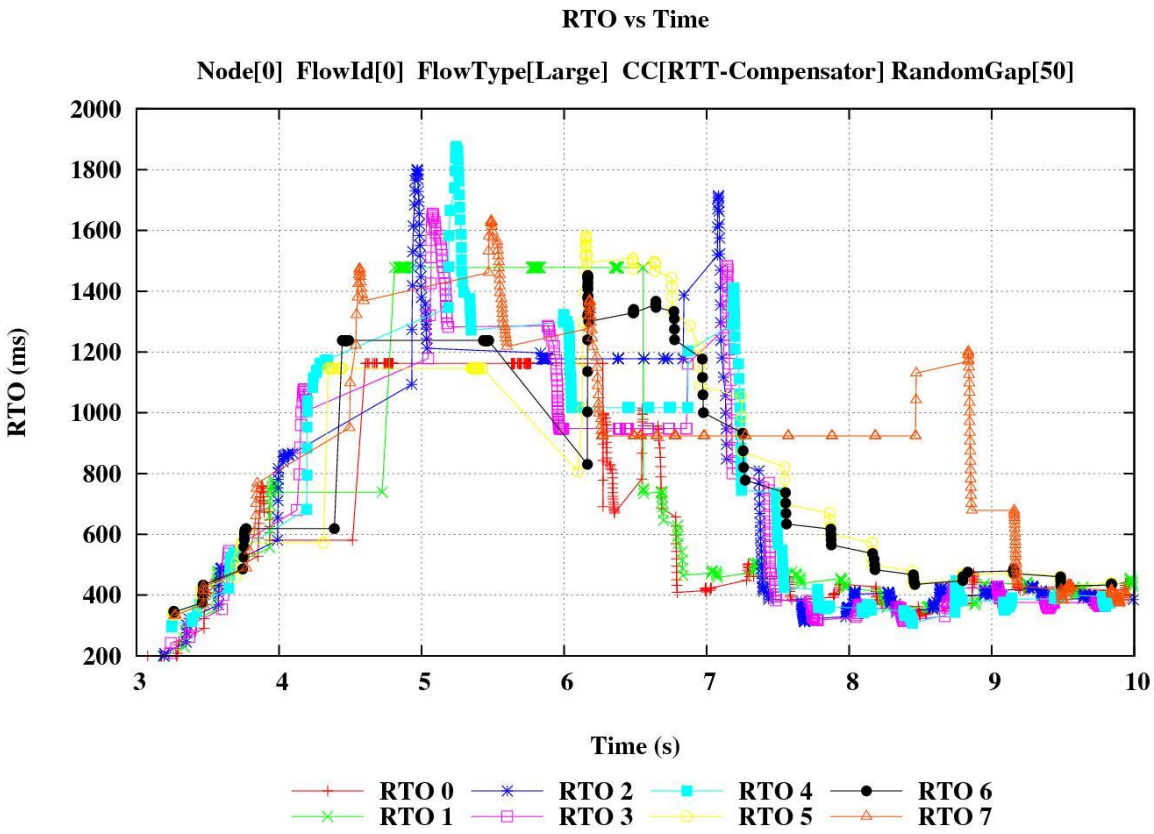
RTT Compensator - Packets per subflow



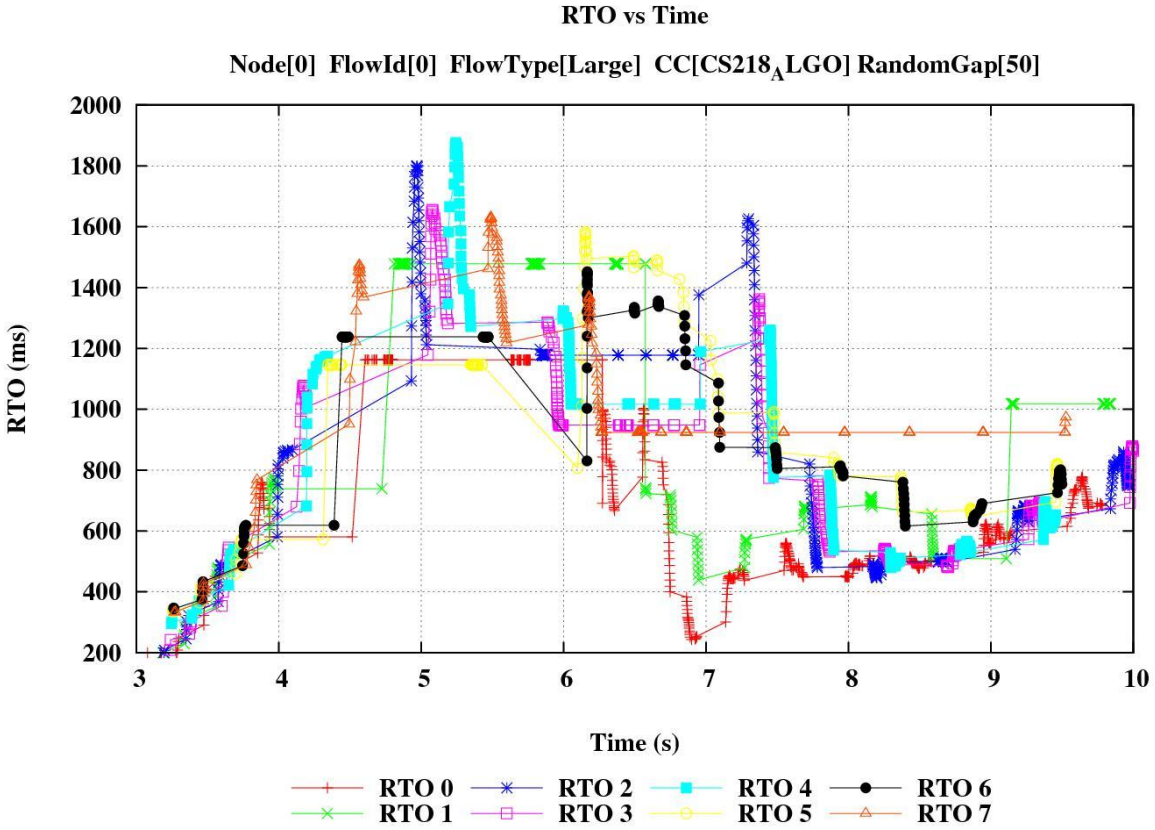
Delay + ACK-count based increase - packets per subflow



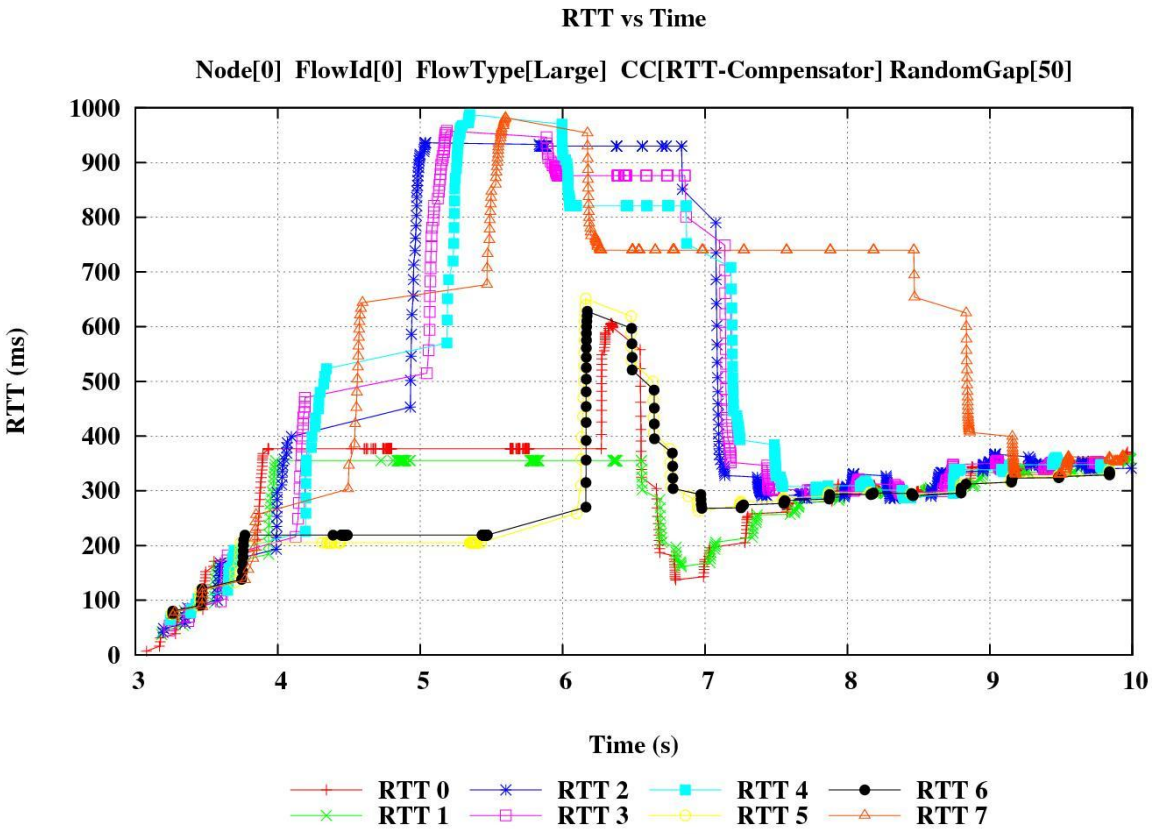
RTT Compensator - RTO vs Time



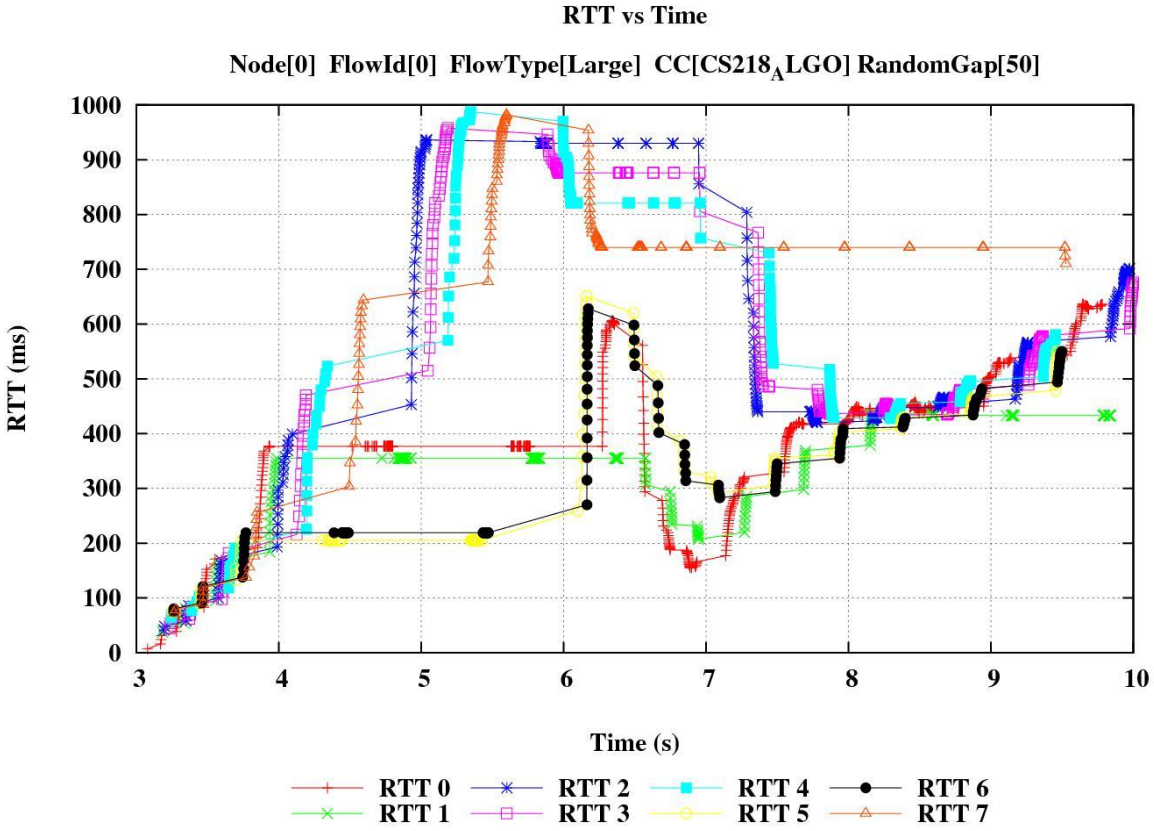
Delay + ACK-count based increase - RTO vs Time



RTT Compensator - RTT vs Time



Delay + ACK-count based increase - RTT vs Time



Future Work and Summary

Conclusion

- Evaluated the limitations of the current MPTCP congestion control algorithms
- Tested new congestion control algorithms in different scenarios
- Our current delay based algorithm increases congestion window rapidly to utilize high speed paths .
- However there is slight increase in RTO and RTT times.

Future Work

- Test variants of TCP congestion control algorithms and adapt them to MPTCP, e.g., TCP New Reno, ATCP, etc.
- Look at larger scale networks consisting of multiple networks of different types, e.g., VANET

References

1. Kheirkhah, M., Wakeman, I., and Parisis, G Multipath-TCP in ns-3. https://arxiv.org/pdf/1510.07721.pdf
2. Coudron, M., Secci, S. (2017). An implementation of multipath TCP in ns3. Computer Networks, Volume 116.
3. Paasch, C., Bonaventure, O. (2014). Multipath TCP. acmqueue, Volume 12.
4. Wischik, D., Raiciu, C., Greenhalgh, A., Handley, M. (2011). Design, implementation and evaluation of congestion control for multipath TCP. NSDI.
5. Hadji, A. (2017). From TCP to MPTCP Brief Explanation. Nov. 5, 2017. https://www.slideshare.net/akremgegawatt/from-tcp-to-mptcp-a-brief-explanation-81616545?next_slideshow=1.
6. Kheirkhah, M. (2014). MPTCP. https://github.com/mkheirkhah/mptcp
7. Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., Handley, M. (2012). How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. NSDI 2012.
8. Handley, M., Wischik, D. Raiciu, C. Coupled Congestion Control for MPTCP. https://www.ietf.org/proceedings/77/slides/mptcp-9.pdf
9. Craig, K. Compound TCP in ns3. https://web.cs.wpi.edu/~rek/Adv_Nets/Fall2014/TCP_Compound_Project.pdf
10. Raiciu, C., Wischik, D., Handley, M. Linked Congestion Control. https://www.ietf.org/proceedings/76/slides/mptcp-8.pdf
11. Lochert, C., Scheuermann, B., Mauve, M. (2007). A Survey on Congestion Control for Mobile Ad-Hoc Networks. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.7225&rep=rep1&type=pdf