TCP Evaluation Suite for ns-3

Dharmendra Kumar Mishra, Pranav Vankar and Mohit P. Tahiliani
Wireless Information Networking Group (WiNG)
NITK Surathkal, Mangalore, India, 575025
dharmendra.nitk@gmail.com, pranavvankar442@gmail.com, tahiliani@nitk.ac.in

ABSTRACT

Congestion Control (CC) algorithms are essential to quickly restore the network performance back to stable whenever congestion occurs. A majority of the existing CC algorithms are implemented at the transport layer, mostly coupled with TCP. Over the past three decades, CC algorithms have incrementally evolved, resulting in many extensions of TCP. A thorough evaluation of a new TCP extension is a huge task. Hence, the Internet Congestion Control Research Group (ICCRG) has proposed a common TCP evaluation suite that helps researchers to gain an initial insight into the working of their proposed TCP extension.

This paper presents an implementation of the TCP evaluation suite in ns-3, that automates the simulation setup, topology creation, traffic generation, execution, and results collection. We also describe the internals of our implementation and demonstrate its usage for evaluating the performance of five TCP extensions available in ns-3, by automatically setting up the following simulation scenarios: (i) single and multiple bottleneck topologies, (ii) varying bottleneck bandwidth, (iii) varying bottleneck RTT and (iv) varying the number of long flows.

CCS Concepts

•**Networks** → **Transport protocols; Network simulations;** •**Computing methodologies** → *Simulation environments;*

Keywords

ns-3, Congestion Control, TCP Evaluation

1. INTRODUCTION

Congestion Control (CC) algorithms implemented in TCP play a vital role in ensuring proper functioning of the Internet. Over the period of time, as CC algorithms continue to evolve, a lot of new TCP extensions are frequently proposed. Evaluating the performance of new TCP extensions is not trivial because there is a lack of agreed set of performance metrics, because of which, each study highlights only a particular aspect of TCP while leaving some of the most important ones. Due to a high volume of research being carried out in this area, there is a need for systematically screening every new TCP extension and identifying the suitable ones for detailed evaluation.

To address this problem, the Internet Congestion Control Research Group (ICCRG) [1] provides information regarding a common test suite for evaluating new TCP extensions. This suite is not framed to *exhaustively* evaluate a new TCP extension; instead, it focuses to help the researchers to *easily* and *quickly* derive initial results of their work.

In this paper, we present an implementation of TCP evaluation suite in ns-3 [2]. We also highlight some additions that were required in existing models of ns-3 to successfully implement the suite. Our implementation can be used to automate the work cycle from setting up the simulation environment to collecting results. Moreover, our implementation provides support for automatically configuring many Internet-like scenarios such as scenarios with single and multiple bottleneck links, scenarios with different traffic mix and scenarios with varying bottleneck attributes. Nevertheless, our implementation is only a part of the evaluation suite designed by ICCRG [1]. Implementing the entire suite requires significant changes to existing models in ns-3 and is beyond the scope of this paper.

The rest of this paper is organized as follows: Section 2 provides a review of similar benchmark proposals for TCP evaluation and their implementation details. Section 3 discusses the design choices and our proposed architecture of TCP evaluation suite in ns-3. Section 4 demonstrates the usage of our suite to compare existing TCP extensions in ns-3 by automatically configuring different simulation scenarios. Lastly, Section 5 summarizes and concludes the paper.

2. RELATED WORK

The design of TCP evaluation suite dates back from a paper in 2007 to an internet draft in 2014, as shown in Table 1. All proposals have been implemented using ns-2 [7]. Although both, proposal 1 and 3, are targeted towards evaluating High-Speed TCPs, each adopts a different approach for implementing it. Proposal 4 is an enhancement of proposal 2, both being the internet drafts. Moreover, they adopt a similar approach for implementation in ns-2; in fact, proposal 4 extends the implementation of proposal 2.

The design and implementation of TCP evaluation suite presented in this paper is partially adopted from the approach followed by proposals 2 and 4. We found that the implementation of TCP evaluation suite in ns-3 is relatively simpler, thanks to the topology helper classes provided.

3. DESIGN AND IMPLEMENTATION

TCP evaluation suite is implemented as a separate model called `tcp-eval`, under the `src` directory in ns-3. This sec-

WNS3, June 15-16, 2016, Seattle, WA, USA

DOI: http://dx.doi.org/10.1145/2915371.2915388

Table 1: Different implementations of TCP Evaluation Suite.

No.	Details of proposals	Tools used
1	Shimonishi, Hideyuki, M. Y. Sanadidi, and Tutomu Murase. "Assessing Interactions among Legacy and High-Speed TCPs." PFLDnet 2007 (2007).	ns-2. Download link [3]
2	Internet draft: "An NS-2 TCP Evaluation Tool", Gang wang, Yong Xia, David Harrison (April 2007)	ns-2. Code was released in two versions. Download link of second version [4]
3	Li, Yee-Ting, Douglas Leith, and Robert N. Shorten. "Experimental evaluation of TCP protocols for high-speed networks." Networking, IEEE/ACM Transactions on 15.5 (2007): 1109-1122.	ns-2. Download link [5]
4	Internet draft: "Common TCP Evaluation Suite", D. Hayes, D. Ros, L. Andrew, S. Floyd (July 2014)	ns-2. Download link [6]

tion describes the core design decisions made for the development of the TCP evaluation suite model, along with several additions that were required in the existing models of ns-3. Table 2 highlights different elements that are supported in our implementation of `tcp-eval`. Figure 1 depicts the interactions among different classes implemented in `tcp-eval`, and with other existing classes of ns-3.

Table 2: Different elements supported by `tcp-eval`.

Topologies	Types of traffic	Performance metrics
Dumbbell (single bottleneck)	Long-lived FTP (TCP)	Aggregate link utilization
Parking lot (multiple bottleneck)	Streaming video (UDP)	Mean queue length
	Interactive voice (UDP)	Packet drop rate

The `tcp-eval` model comprises three primary classes:

3.1 ConfigureTopology

This class is used for configuring the simulation parameters such as setting bottleneck bandwidth and bottleneck delay, setting the parameters for Random Early Detection (RED) algorithm [8], etc. It acts as a base class to configure parameters for dumbbell and parking lot topologies.

3.2 CreateTraffic

This class generates different traffic patterns for the simulation, as listed in Table 2.

Long-lived FTP traffic: It runs on top of TCP and is generated by using the `BulkSend` Application provided in ns-3. It generates a stream of packets, each of size 512 bytes, till the end of simulation. Moreover, this traffic can be generated in forward, reverse and cross directions. Cross FTP traffic is generated only for parking lot topology.

Streaming video traffic: It runs on top of UDP and is generated by using the `OnOff` Application provided in ns-3. The default packet size is set to 840 bytes and the streaming rate is set to a default value of 640 Kbps [4]. The packet size and streaming rate can be explicitly configured by the user through command line arguments. This traffic can be generated in forward and reverse directions.

Interactive voice traffic: This traffic also runs on top of UDP and is generated by using the `OnOff` Application provided in ns-3. The default packet size is set to 172 bytes and the data rate is set to a default value of 64 Kbps [4]. This is a two-way traffic between the caller and the callee.

3.3 EvalStats

This class is used for collecting post simulation results and store them in files, which are later used to plot graphs. All the performance metrics listed in Table 2 are calculated in this class.

Aggregate Link Utilization: This metric specifies the ratio of current network traffic to the maximum available bandwidth. It is important for a new TCP extension to maximize the bandwidth utilization, while ensuring fairness with other TCP flows.

Mean Queue Length: Maintaining a steady queue length is important to avoid variations in delay. Large variations in delay affect the user perceived application behaviour, especially in the case of interactive voice applications and streaming applications. This metric is important to analyse the stability of a new TCP extension.

Average Packet Drop Rate: This metric is crucial to analyse the performance of TCP in the presence of bursty background traffic. High packet drop rate hurts the performance of time sensitive traffic like Google search, etc.

3.4 Other Classes in `tcp-eval`

TrafficParameters: This class provides setters and getters for configuring the traffic related parameters like:

- Number of forward FTP flows
- Number of reverse FTP flows
- Number of cross FTP flows
- Number of two-way voice flows
- Number of forward streaming flows
- Number of reverse streaming flows

DumbbellTopology: This class sets up a dumbbell simulation scenario, and is placed in a file called `dumbbell-topology.cc` in `tcp-eval` model. First, it creates a dumbbell topology by invoking `PointToPointDumbbellHelper` class which is already available in ns-3. Next, it obtains the simulation parameters from `ConfigureTopology`, generates the traffic using `CreateTraffic`, and finally calculates and stores the results using `EvalStats`. Figure 2 shows user's interaction with `tcp-eval` model for simulating dumbbell

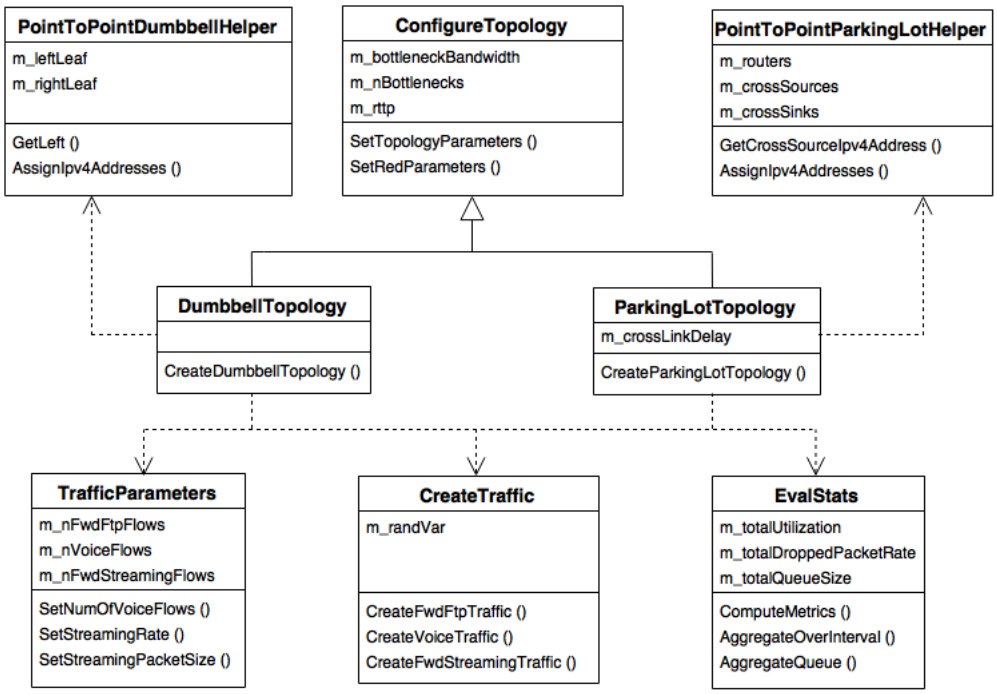


Figure 1: Class diagram of TCP Evaluation Suite in ns-3.

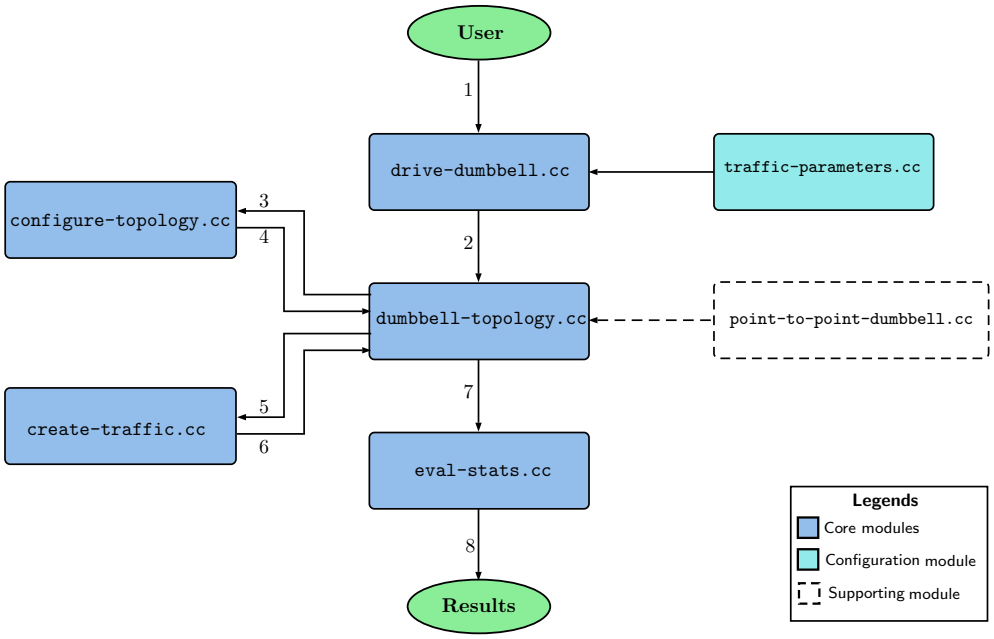


Figure 2: User interaction diagram of `tcp-eval` for dumbbell scenario.

scenarios. `drive-dumbbell.cc` creates an object of `TrafficParameters`, before creating an object of this class.

ParkingLotTopology: While implementing this class we found that a helper for creating multiple bottleneck topology, like parking lot topology, is not available in ns-3. Hence, we implemented a class called `PointToPointParkingLotHelper` and included it in the `point-to-point-layout` model of ns-3. This helper is a stand-alone implementation and not closely linked to `tcp-eval` model. It can be used to simulate parking lot scenarios, even without the `tcp-eval` model.

The functionality of `ParkingLotTopology` class is equivalent to that of `DumbbellTopology`, except that it is responsible to set up a parking lot simulation scenario. This class is placed in a file called `parking-lot-topology.cc` in `tcp-eval` model. Figure 3 shows user's interaction with `tcp-eval` model for simulating parking lot scenarios. `drive-parking-lot.cc` creates an object of `TrafficParameters`, before creating an object of this class.

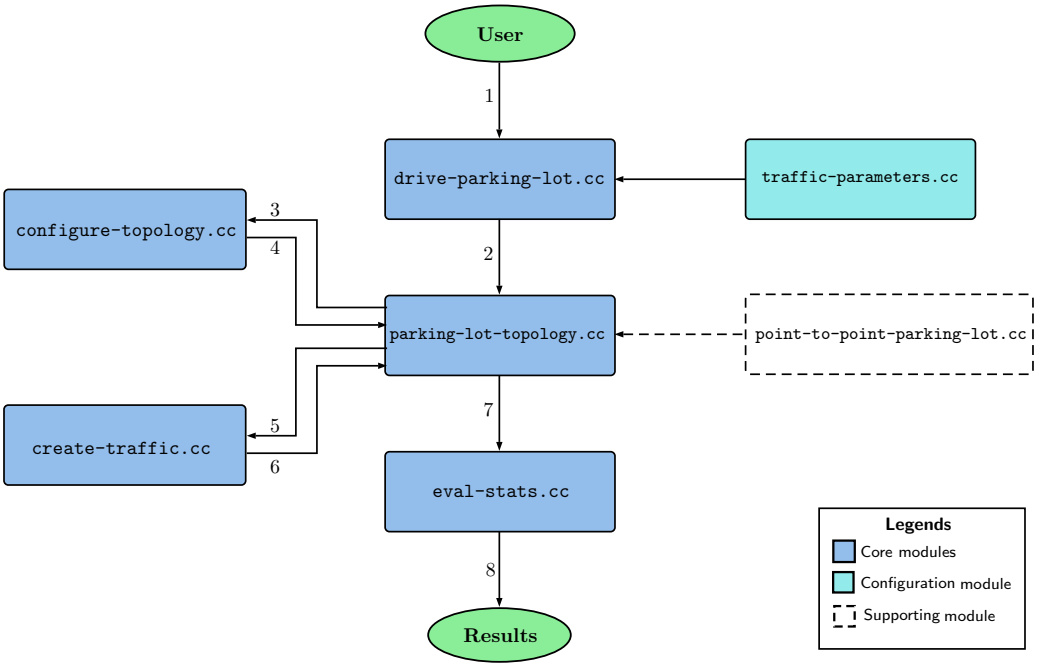


Figure 3: **User interaction diagram of `tcp-eval` for parking lot scenario.**

4. NS-3 TCP EVALUATION USING TCP-EVAL

In this section, we compare the performance of five TCP extensions available in ns-3 by using our implementation of the suite. We provide a set of six shell scripts that run a series of simulations by varying the following three parameters: bottleneck bandwidth, bottleneck RTT and the number of forward FTP flows in two topologies: dumbbell and parking lot. Once the results are collected in respective files for every scenario, shell scripts convert those textual results into graphical form. Further, a PDF file containing graphs for each scenario is created automatically if LaTex is installed.

As suggested by ICCRG, we randomize the start times of all traffic flows to model the real traffic behaviour. Moreover, our implementation provides support for simulating RED algorithm [8] at bottleneck routers. We have used the traditional droptail mechanism in our scenarios, however, to keep the analysis simple. The bottleneck routers can be easily configured to use RED algorithm by passing command line arguments. No further changes are required.

Table 3: **Default simulation parameters.**

Simulation Parameters	Values
Bottleneck bandwidth	10 Mbps
Round Trip Time	80 ms
Number of forward FTP flows	5
Number of reverse FTP flows	5
Number of voice flows	5
Number of forward streaming flows	5
Number of reverse streaming flows	5
Simulation time	100 seconds
Streaming rate	640 Kbps
Streaming packet size	840 bytes

4.1 Varying Bottleneck Bandwidth

In this scenario, the bottleneck bandwidth is varied from 1 Mbps to 100 Mbps. Other parameters shown in Table 3 remain fixed. For every TCP extension, simulation runs till the specified duration for a collection of bandwidth values.

Varying the bottleneck bandwidth provides an estimate of TCP's performance under the same traffic load, but different bottleneck capacity. Figure 4 and 5 depict the simulation results obtained by varying bottleneck bandwidth in dumbbell and parking lot topologies, respectively. It can be observed that the bottleneck link occupancy in both the figures is close to 100% for all TCP extensions when the bandwidth values are relatively low, and it gradually decreases with the increase in the bandwidth. It is important for any TCP extension to adapt its congestion window to ensure that bottleneck bandwidth remains fully utilized.

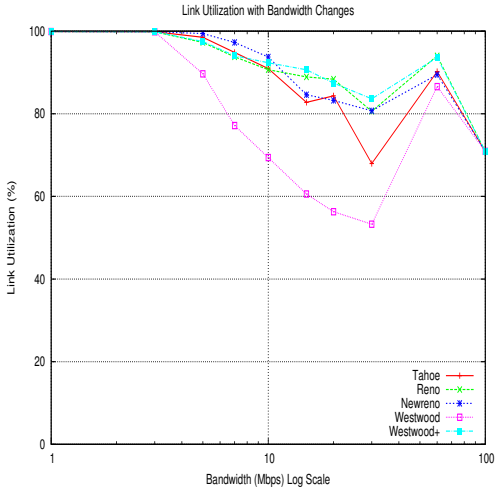
4.2 Varying RTT

In real internet scenarios, the hosts can be distributed over the large span of geographical area. Such hosts can be modelled by varying the propagation delays between the nodes in the simulation environment. Hence, in this scenario, we change the RTT values between the bottleneck routers from 10 milliseconds to 1 second, while keeping other values fixed as shown in Table 3.

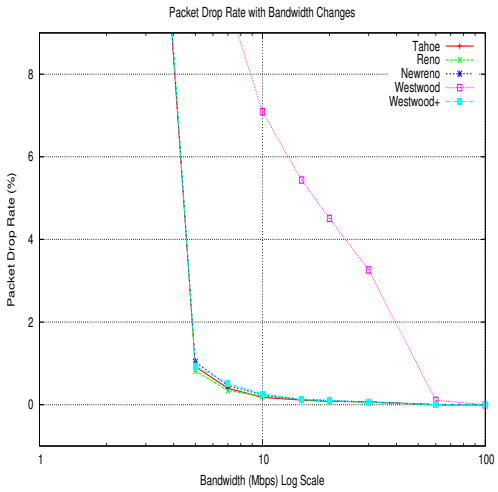
Figure 6 and 7 depict the simulation results obtained by varying the RTT values between the bottleneck routers in dumbbell and parking lot topology. The results clearly depict the performance of each TCP extension, and how its behavior differs from other TCP extensions.

4.3 Varying the Number of FTP Flows

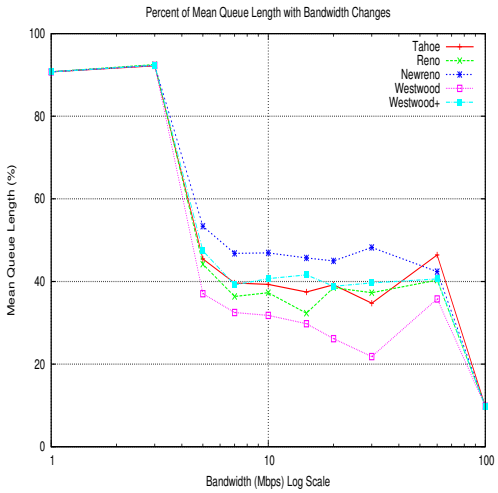
This scenario is designed to test the performance of TCP extensions by varying the traffic load. We run a series of simulations by varying the number of forward FTP flows from 1 to 100. Other simulation parameters listed in Ta-



(a) **Aggregate bandwidth utilization.**

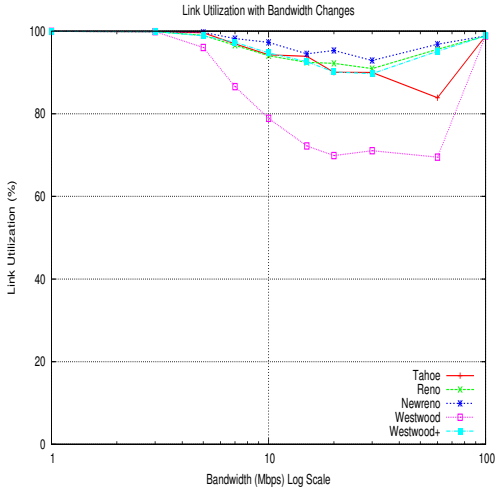


(b) **Packet drop rate.**

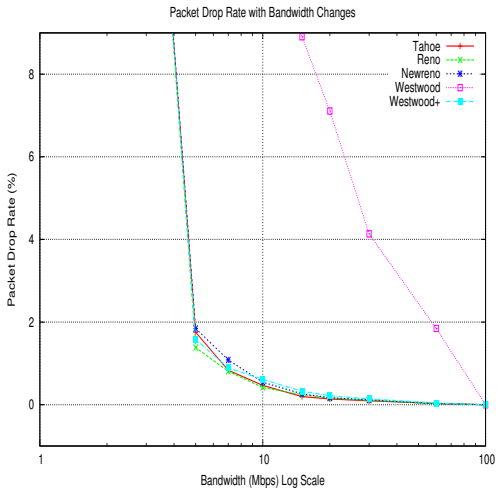


(c) **Mean queue length.**

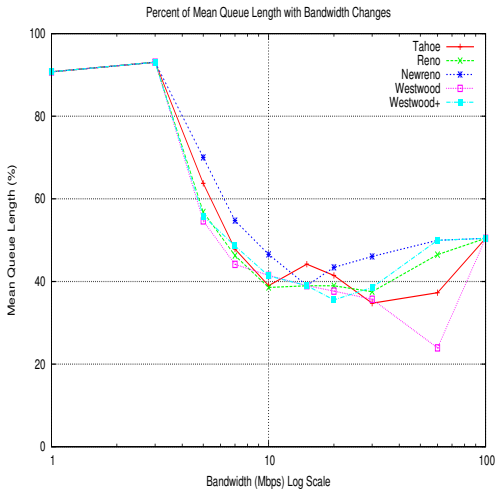
Figure 4: TCP performance results for changing bandwidth for dumbbell topology.



(a) **Aggregate bandwidth utilization.**

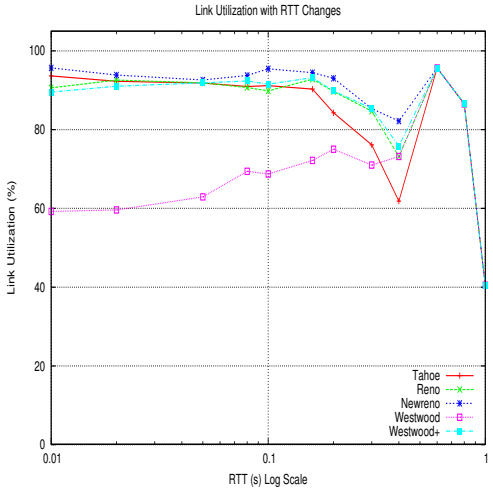


(b) **Packet drop rate.**

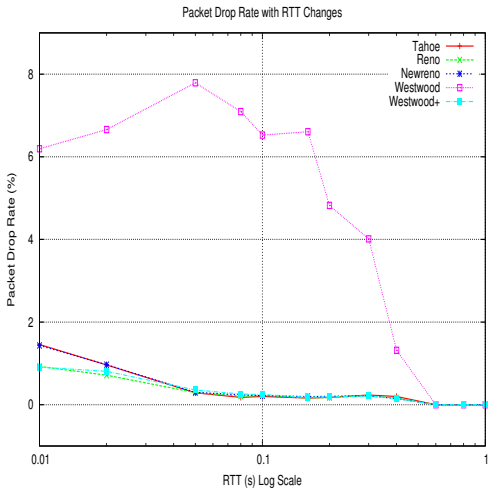


(c) **Mean queue length.**

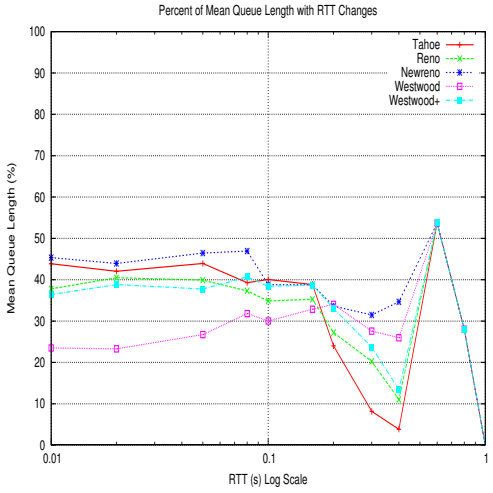
Figure 5: TCP performance results for changing bandwidth for parking lot topology.



(a) **Aggregate bandwidth utilization.**

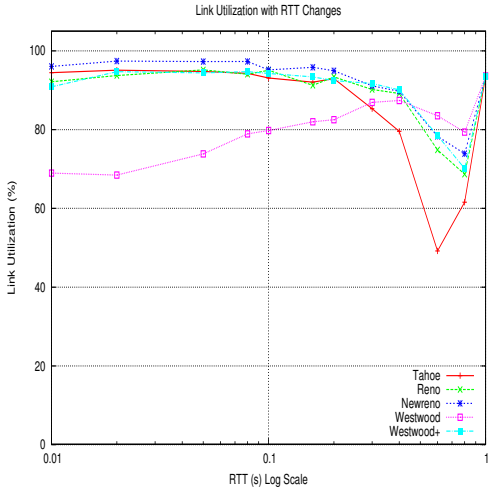


(b) **Packet drop rate.**

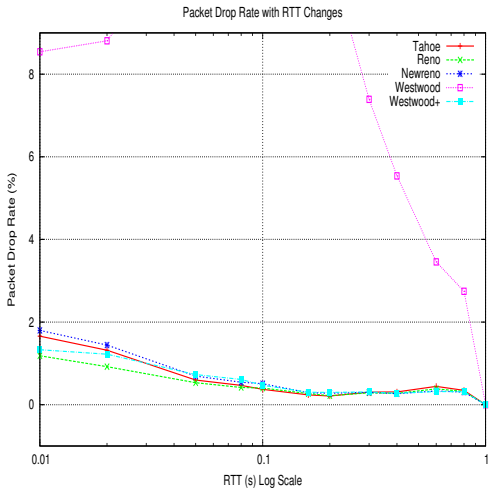


(c) **Mean queue length.**

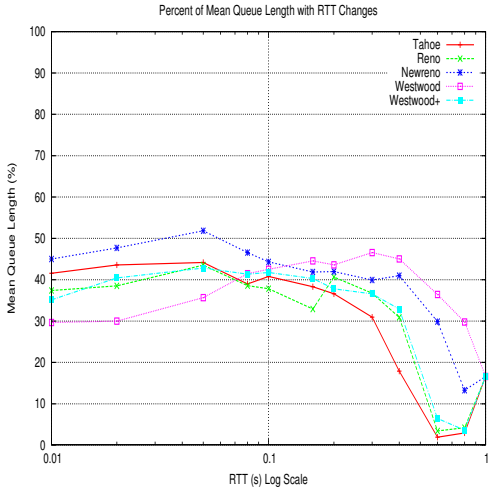
Figure 6: TCP performance results for changing RTT for dumbbell topology.



(a) **Aggregate bandwidth utilization.**

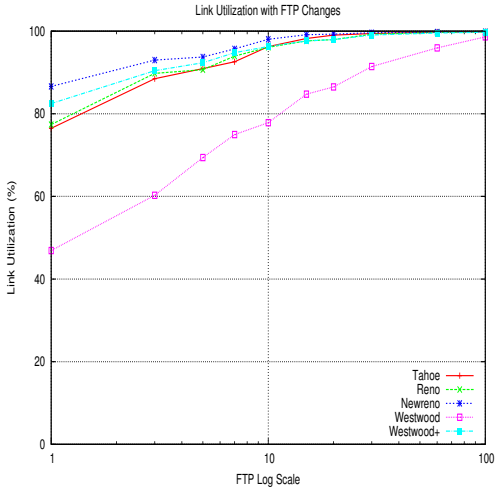


(b) **Packet drop rate.**

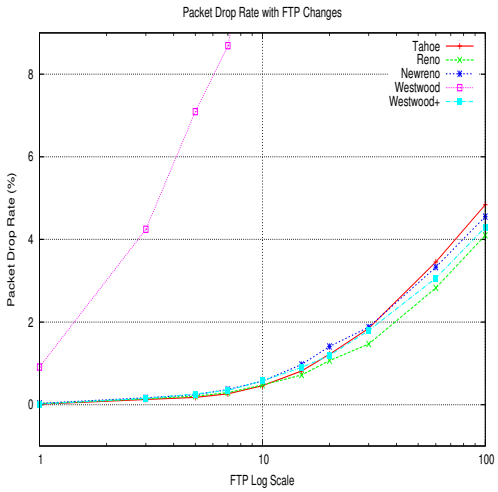


(c) **Mean queue length.**

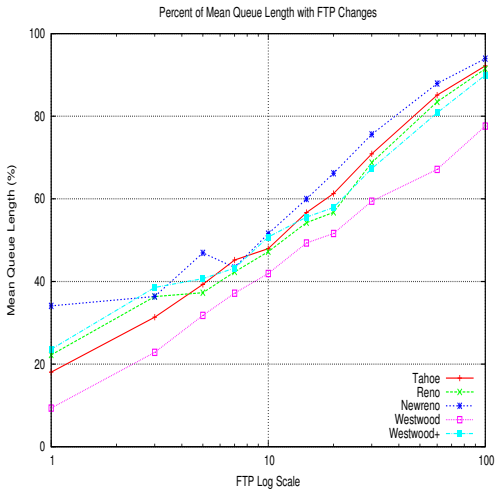
Figure 7: TCP performance results for changing RTT for parking lot topology.



(a) **Aggregate bandwidth utilization.**

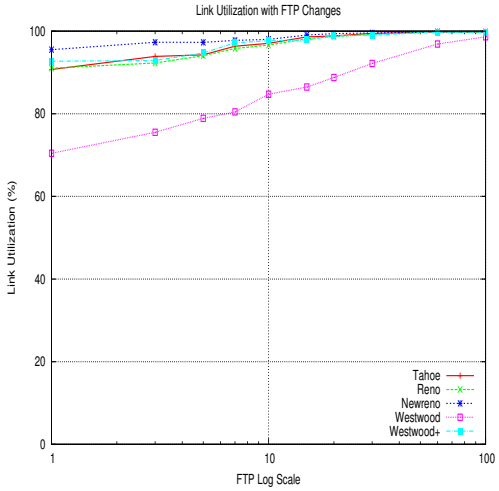


(b) **Packet drop rate.**

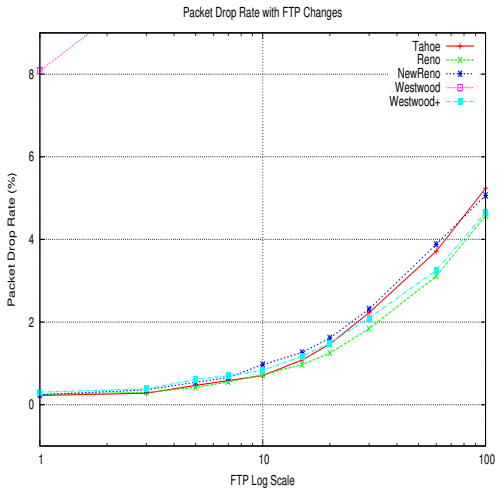


(c) **Mean queue length.**

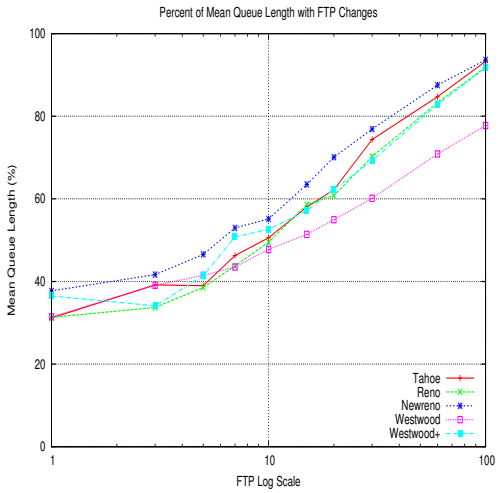
Figure 8: TCP performance results for changing FTP flows for dumbbell topology.



(a) **Aggregate bandwidth utilization.**



(b) **Packet drop rate.**



(c) **Mean queue length.**

Figure 9: TCP performance results for changing FTP flows for parking lot topology.

ble 3 remain fixed. Figure 8 and 9 depict the simulation results obtained for dumbbell and parking lot topology respectively. The bottleneck link utilization remains close to 90% for small number of FTP flows, and as expected, gradually increases with the increase in traffic load.

All graphs presented for this scenario and previous two scenarios are automatically generated using the shell scripts. Instructions to reproduce these results are provided in the Appendix.

5. CONCLUSION AND FUTURE WORK

In this paper, we present an implementation of TCP evaluation suite as a separate model in ns-3. We provide the implementation details for every class and highlight the interactions among them. Additionally, we demonstrate the usage of our TCP evaluation model by comparing five TCP extensions available in ns-3 in benchmark scenarios stated in the internet drafts. The results are collected for aggregate link utilization, mean queue length and packet drop rate, and the statistics are generated in textual and graphical formats.

We plan to further extend our implementation and add other features suggested in the draft of ICCRG. We have completed porting Tmix traffic generator to work with the latest version of ns-3. The next step would be to integrate it with our implementation and provide flexibility in terms of using realistic traffic patterns for evaluating the performance of TCP extensions.

6. ACKNOWLEDGMENTS

We would like to acknowledge the support of Gopika Pai, H. J. Bhargav, Pratheek B., Sourabh S. Shenoy for helping in the implementation of EvalStats class. Further, we would like to acknowledge Radhesh Anand for correcting the parts of this paper.

7. REFERENCES

[1] D. Hayes, D. Ros, L. L. H. Andrew, and S. Floyd. Common TCP Evaluation Suite. Internet-Draft draft-irtf-iccrg-tcpeval-01, Internet Engineering Task Force, January 2015. Work in Progress.

[2] Network Simulator 3. https://www.nsnam.org, 2016.

[3] H. Shimonishi, M. Y. Sanadidi, M. Gerla, C. Marcondes, and P. Vasu. TCP Evaluation Suite. http://nrlweb.cs.ucla.edu/tcpsuite/index.html, 2007. Evaluating New Congestion Control Schemes and Its Impact on Standard TCP NewReno.

[4] An NS2 TCP Evaluation Tool. https://sourceforge.net/projects/tcpeval, 2007.

[5] Y. Li, D. Leith, and R. N. Shorten. Hamilton Institute TCP Evaluation Suite. http://www.hamilton.ie/net/eval/hi2005.htm, 2007.

[6] D. Hayes, D. Ros, L. Andrew, and S. Floyd. Common TCP Evaluation Suite. https://bitbucket.org/hayesd, July 2014.

[7] Network Simulator 2. http://www.isi.edu/nsnam/ns, 2016.

[8] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *Networking, IEEE/ACM Transactions on*, 1(4):397–413, 1993.

APPENDIX

In this section, we provide additional details about reproducing the simulation scenarios described in this paper.

The latest version of ns-3 at the time of writing this paper is ns-3.24 and the same has been used for implementing the TCP evaluation suite. The modified ns-3.24 that contains our TCP evaluation suite implementation can be obtained from here[1]. All the results obtained from our suite get stored in a new directory called `tcp-eval-results` in `ns-3.24` directory.

TCP evaluation suite is implemented in `src/tcp-eval` and its the classes can be found at `src/tcp-eval/model`. The helper for point to point parking lot topology is located at `src/point-to-point-layout/model/point-to-point-parking-lot{.h, .cc}`. `drive-dumbbell.cc` and `drive-parking-lot.cc` are available at `src/tcp-eval/examples`. Several command line arguments can be passed to both; more details can be found in the respective files. Lastly, six shell scripts are provided in `ns-3.24` directory that reproduce the graphs presented in Section 4 of this paper. Each shell script produces three graphs, and if LaTex is found installed on the machine, it places the graphs in respective PDF files on successful completion.

[1]https://github.com/dharmendra-mishra/wns3-2016