Evaluation of PI Controller Queue Management Algorithm using ns-3

Mini Project Report
Submitted
by

Pranab Nandy
(202CS019)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE -575025

May 15, 2021

Acknowledgements

In my journey towards the completion of my mini-project, I was assisted and guided by numerous people. Words are insufficient to express my sincere gratitude towards my supervisor, Dr. Mohit P. Tahiliani, of the Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal, India, for his constant guidance, support and encouragement. He has given me a lot of freedom in my research and allowed me to work on the problems of my interest. His passion for research showed me how academia can be exceedingly rewarding and great fun. I have learnt a lot and still have plenty to learn from him.

Last but not the least, we would like to extend our warm regards to our families and peers, especially Sushma Meena and Aditya Chowdhury, who have kept supporting us and always had faith in our work.

Abstract

Congestion in Networks is a very important and serious problem. To maintain the stability of the network, people have been working on congestion control algorithms. Memory price has been decreasing since last decade. Due to this, routers are having large sized buffers. These large sizes buffers lead to high queuing delays. This problem is known as the Bufferbloat problem. It is the undesirable latency coming from router as it is buffering excessive data. Internet performance gets affected due to the Bufferbloat problem. We have to use some techniqueue to solve the high queueing delay problem provided that throughput is also high.

The Queue Management Method deployed inside the routers is an important factor in the congestion control study. Active Queue Management (AQM) has been proposed to maintain the queue for avoiding the congestion inside the network. AQM algorithms provide the solution to control the queuing delay by controlling the queue occupancy to the desired target. PI Controller is a well known AQM technique which is being revisited by the research community. In this project, we try to evaluate PI controller's effectiveness to provide high resource utilization, identifying and restricting disproportionate bandwidth usage. Based on modern day requirements, we designed the network topology in ns-3 and try to cover different network situations to validate its effectiveness. At last we used AQM test suite for PI model validation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Network Congestion is one of the most important problems in the Internet. Most of the routers and network link play a passive role in congestion control. Congestion in the network indicates the reduction of quality of services that occurs when a network link or node is carrying huge amount of data beyond its capacity. Typical effects of congestion are huge packet losses, high queueing delay or the blocking the establishment of new connections [8]. At very high traffic in network, performance collapses completely and almost no packets are delivered to the destination.

Congestion control tries to modulate the traffic which are coming into a network for avoiding congestive that occurs from the over subscription for high bandwidth by the users. This is generally conducted by decreasing the rate of packets send by the senders. It stops the senders from flooding in the network by restricting its sending rate. On the other hand the flow control prevents the sender from overwhelming the receiver host by decreasing the congestion wondow size (Cwnd).

Queue management algorithms are designed for managing the router buffers by dropping the incoming packets whenever it is necessary to drop for better performance. These algorithms are classified into 2 parts which are Passive Queue Management (PQM) and Active Queue Management (AQM). PQM algorithms are drop-head, random drop , drop-tail etc. The PQM suggests that do not drop or mark any packets for managing the buffer until it is full or half full. This passive approach leads to performance issues in the network. There is a situation in the network where either a single flow or a group of flows consume the entire buffer of the queue and left only a little or no space for other flows while we use Passive Queue Management. It occurs when the bit-torrent packets take most the space in the router buffer and leaves little space for HTTP flows.

There is another problem called Full queue problem which decreases the burst consuming capability of the queues due to PQM algorithms which mark or drop packets only when the queue is full or half full. It is known as bufferbloat problem [2] which occurs due to high queueing delay in the network. AQM algorithms on the other hand drop the packets before the queue becomes full and take some action to measure or resolve the issues by marking the ECN bit. However, the main problem is that the deployment of AQM algorithm in the Internet has been a critical. The AQM algorithms require to set some parameters manually like α and β in PI controller. We know that

wrong settings of the manual parameters of AQM algorithms generally lead to worst results. In some network condition it might work better but when the network situation changes the it can not able to perform optimally. Hence, there are high interest in the researches to design and devoloping an efficient and easy to deploy AQM algorithms.

The ns-3 source code was already available for PI controller [4] but it was outdated. Previously, the network traffic was completely different from the today's internet traffic. So we have updated the old PI Controller algorithm to according to latest internet traffic. After revising it to latest algorithm which can fit into ns-3, we performed certain kinds of validation. The validation has done to confirm after shifting to latest the ns-3, we did not break the algorithm logic. We have proved that our algorithm is working properly as expected. Old source code will go through several modifications and adding up new feature to become to beat the modern network traffic. After completing the implementation part of the source code, we performed some kind of evaluation. So our first basic evaluation conducted using congestion control topology like dubmbell topology with 2 sources and 2 routers. At the last stage of our project, we performed a thorough analysis of the PI controller algorithm by using an AQM Evaluation Suite. After the detailed analysis of our algorithm we have planned align with latest ns-3 module for research and development purpose.

Chapter 2

Background and Related Work

2.1 Choice of PI Controller Algorithms

AQM algorithms gives a congestion solution by considering the queue state and dropping rate of the packets in the router buffer. The classification of AQM algorithms is generally considered based on some parameters like queue length (e.g. RED, PI, etc), packet loss (e.g. BLUE etc), queue delay (e.g. CoDel, PIE, etc). The algorithm which uses the same congestion parameter differ from others in the way by which it decides how packet drop event will occur like the drop probability in PI Controller could be a function of the instantaneous queue length, whereas it is a function of common queue length in RED algorithm [3].

We decided to evaluate PI algorithm in ns-3 because lots of recent research considering its advantages to new AQM algorithms. Previosuly researcher had implemented the model of PI i,e Proportional Integral. However, its code is back dated by comparing it with the recent C++ compilers in ns-3. And also an AQM Evaluation Suite has designed for ns-3 in recent years which allows researchers to verify the functionality of AQM algorithms based on the rules provided in RFC 7928. Thus, considering a model for PI in ns-3 might be helpful for researchers to review the performance of AQM algorithms before studying more recent algorithms like PI^2 which is an advance on PIE algorithm [8]. Moreover, ns-3 traffic control system does not have any instantaneous queue length based AQM algorithms like PI controller for AQM evaluation. As we said that PI algorithm slightly differs from RED while both are supported queue length. Finally, we note that the performance of PI had been evaluated within some set of network condition. But in this project we will evaluate its performance with all other AQM algorithm through AQM test suite.

2.2 Overview of ns-3

ns-3 is an network simulator that has support for network protocol simulations like TCP, UDP, internet protocol etc. It has been widely used open source network simulator in last decade and has taken the attention of many researchers who are doing research

on networking domain due to its built-in support of many network protocols. The main reason for selecting ns-3 over other netowork simulator like ns-2 , NeST etc is that it provides a Linux-like control subsystem like it has traffic control subsystem for AQM. Whereas if we consider NeST, it does not provide such functionality and flexibility. According to me, ns-3 is that the only network simulator that has a Linux-like control subsystem [1]. This kind of functionality enables a new oppurtunity for students and reseachers to consider ns-3 as more stable and flexible than other network simulators for verifying and implementing the performance of queue mangement algorithms.

Another reason of considering of ns-3 is that the kind of support it has regarding the distributed execution. Parallel simulations are really very helpful to execute the very large model by utilizing the power of the multi-core architecture through high performence computing like GPU [2]. ns-3 enables the running of one simulation on multiple processors by considering the concept of standard Message Passing Interface (MPI) . This not only improves the speed of execution but also increases the flexibility for network simulations. Since our implemented model is designed such that it should be an integral a part of ns-3, its parallel and distributed simulation functionality are the main factor for good performance of the model while doing a high network simulation experiment.

2.3 Traffic Control Layer in ns-3

Traffic control layer in ns-3 follows same kind of implementation of control subsystem that Linux network stack contains in the system. This feature is implemented as a new layer called Traffic Control Layer which stands above the NetDevice and below the IPv4Interface or Ipv6Interface along with AddressHelper. Queue management algorithm are one of the most important part of this traffic control layer. Traffic Control Layer gives an abstract base class called QueueDisc which is used as parent class to derive any new class to implement specific queue mangement algorithm like RED, BLUE and PIE [8]. The following abstract methods which are present in QueueDisc are need to be implemented by every queue management algorithm:

- bool DoEnqueue (Ptr<> item): this method is called on arrival of every single packet and is responsible to enqueue the incoming packet based on some condition.
- Ptr<> DoDequeue (void): this method is invoked to dequeue the packet from router buffer.
- Ptr<> DoPeek (void) const: this method is called to peek the front item from the router buffer.
- bool CheckConfig (void) const: this method is invoked to test whether the configuration of the queue management algorithm is correct or not.
- void InitializeParams (void) : this method is called to initialize the differnt parameters which is required by the queue discipline.

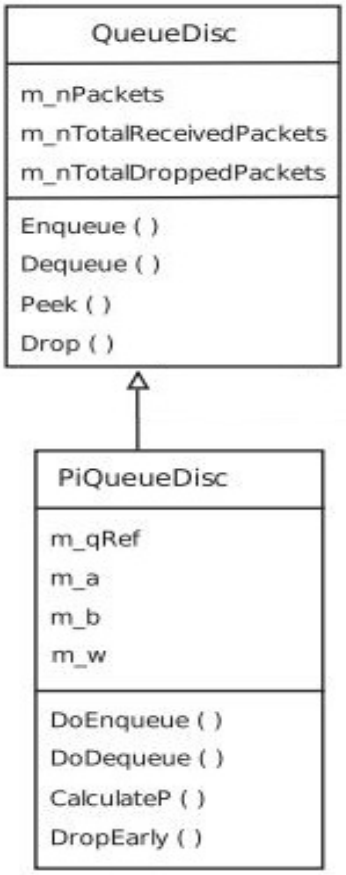


Figure 2.1: Class diagram for PI model in ns-3 [1]

Besides these, every queue management algorithm implements additional methods as per the requirements of the algorithm. Our PI model has also followed the same approach in ns-3. Since PI controller has been added as different queue management algorithm within the control module, the features of existing queue disciplines in ns-3 will not get effected.

PI model has inherited the QueueDisc class. Here we also used the parent class method during the queue management. At the same time, PI model added some more new method like CalculateP() , DropEarly() etc. We explained the functionality of each method in more details of this traffic control layer's class PiQueueDisc in this report.

Chapter 3

PI model in ns-3

3.1 Overview of PI Controller

A dynamic model of TCP behaviour was designed with the help of fluid flow analysis in congestion aviodance state [5] . Here we used a simplified version of that devoloped model which ignores the TCP timeout functionality. This model tries to relate the average value of the main network variables and is described by the subsequent coupled, nonlinear differential equations:

$$\frac{dW(t)}{dt} = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t)} p(t-R(t)) \tag{3.1}$$

$$\frac{dq(t)}{dt} = \frac{N(t)}{R(t)} W(t) - C \tag{3.2}$$

where

Table 3.1: System Parameters

Notations	Explanation
W	TCP Congestion Window Size (packets or bytes)
q	Current Queue Length (packets or bytes)
N	No of TCP flows
C	Link Capacity (packets/sec)
R	Round-trip time (sec)
s	A Complex number frequency parameters

The Fig 3.1 shows a non-linear dynamic system for TCP/AQM. To convert the non-linear system into a lineared model, a number of differentaition on TCP congestion window and quequ length were performed. Then Laplace tranformation were applied on the differented equations. Finally the linearized control system of AQM [5] was

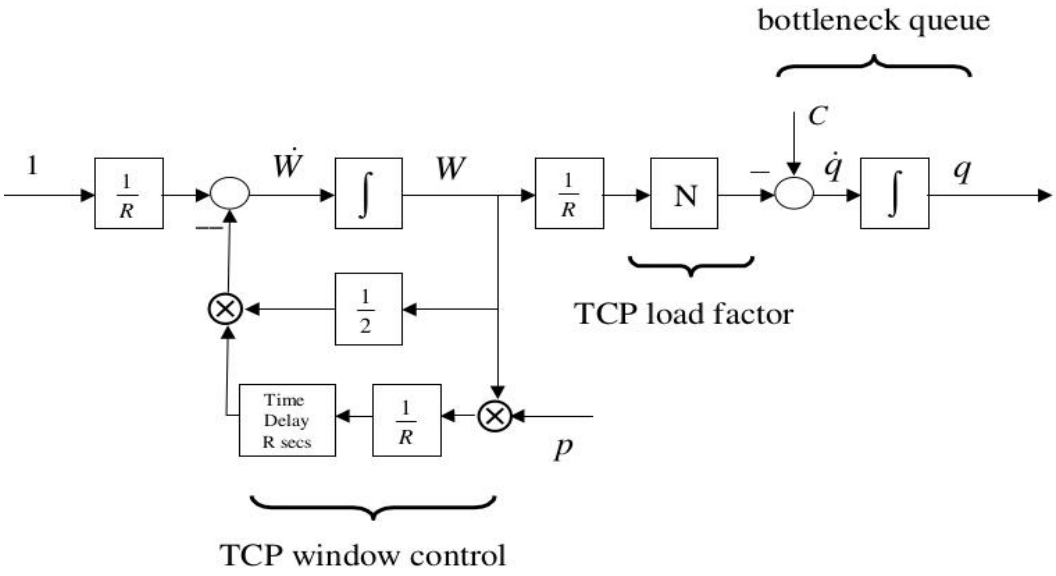


Figure 3.1: Block-diagram of TCP's congestion avoidance flow control mode [5]

obtained in Fig 3.2 which are used as a reference system for PI controller.

In Figure 3.2 the transfer function C(s) denotes an AQM control strategy such PI. The plant transfer function P(s)=$P_{tcp}(s)P_{queue}(s)$. where $P_{tcp}(s)\&P_{queue}(s)$ are two poles are laplace equation.
Here e^{-sR_0} is the laplace time delay.

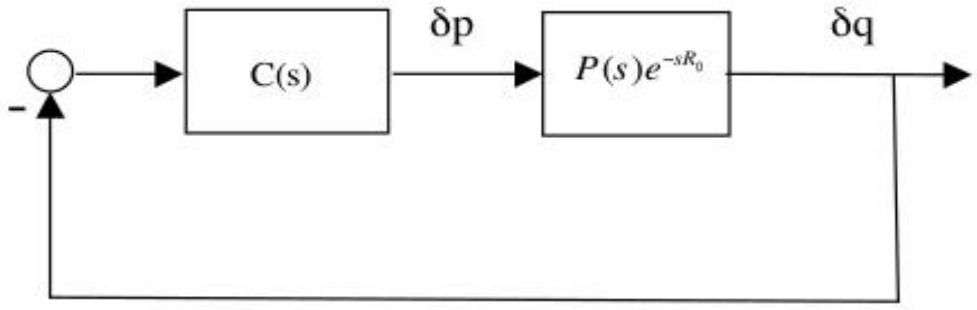


Figure 3.2: AQM as feedback control [5]

3.2 Implementation of PI Controller in ns-3

PI algorithm uses the concept of Proportional Integral [4] to maintain the queue length at routers with lots of flexibility. It allows the user to mention the desired queue length and accordingly it tries to maintain it by updating the packet drop probability at a certain interval. PI changes queue length from the packet drop probability and so it does not require a low pass filter which are mainly used by RED algorithm. The

most important component of PI Controller is Drop Probability Calculation for droping a packet earlier. PI algorithm randomly drops the incoming packets based on the return value of drop early function. And PI algorithm calculates the packet drop probability (p) at a regular interval where sampling frequency is used as W . The parameters involved in the calculation of drop probability are:

- qlen : current queue length (in packets or bytes).
- qold : queue length during the previous sample (in packets or bytes).
- qref : desired queue length.
- pold : drop probability during the previous sample.
- α and β : scaling factors.

p is calculated as:

$$p = \alpha * (qlen - qRef) - \beta * (qOld - qRef) + pOld \tag{3.3}$$

The queue reaches a steady state when α * (q - qRef) - β * (qOld - qRef) becomes zero, implying that the queue length has reached the desired value.

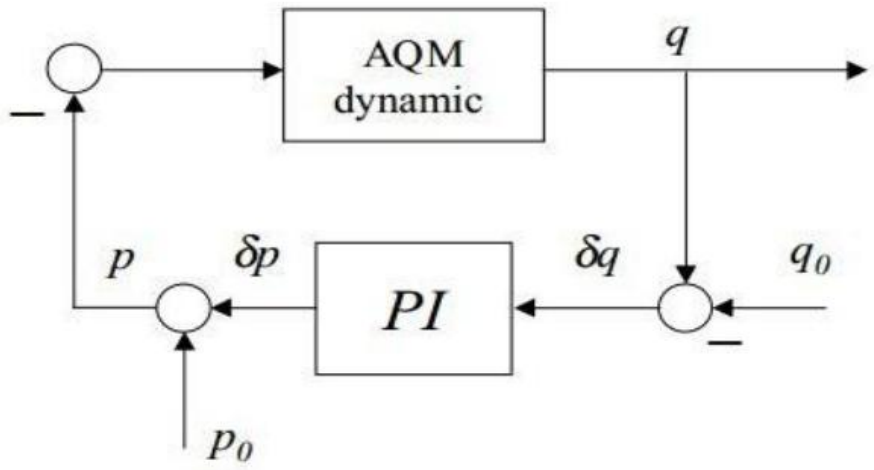


Figure 3.3: Implementation of PI controller emphasizing the role of operating point q_0 [5]

PiQueueDisc class is inherited from QueueDisc class which is the parent class and it implements all the methods. Additionally it implements the following two most important methods that are only related to the PI Controller [1] Model like CalculateP() and DropEarly(). Fig. 3.5 shows the interaction amoung different methods that are present in PiQueueDisc. The PI queue management algorithm drops the packet during the enqueue time, so its random dropping functionality for packet dropping is present in DoEnqueue() method. DoEnqueue() invokes DropEarly() method, which decides

whether the incoming packet should be enqueued or dropped based on some certain condition. The decision to drop a packet is considered by comparing the value of drop probability p with a random value u obtained from UniformRandomVariable class in ns-3. The packet is enqueued if p < u, otherwise packet will be dropped.

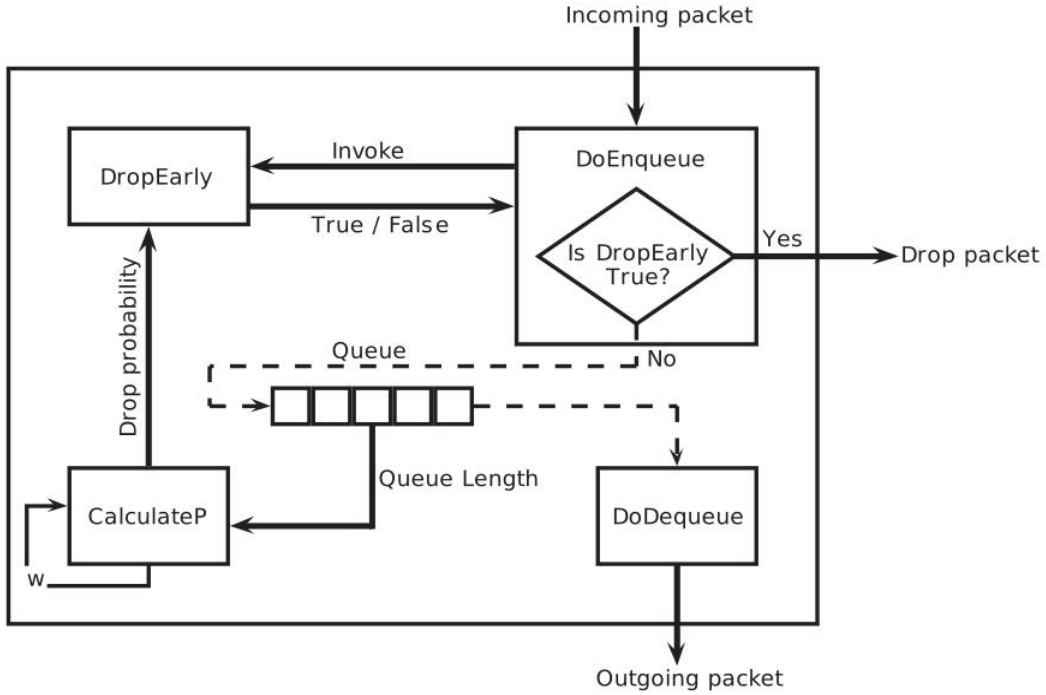


Figure 3.4: Interaction between different methods of PI algorithm [1]

CalculateP() method checks the current queue length, calculates the drop probability (p) and perform an updation on the old queue length value. It is called at certain time intervals based on the sampling frequency w. Among all the parameters, q len and q old are updated by the PI queue management algorithm and others parameters are being configured by the user.

Previously the code was implemented by Viyom Mittal. But that code was not compatable with current version of ns-3. I have made several changes in the test-suite to make compatible with current version. I have also added PI support in pie-example.cc file to compare its performance with PIE algorithm. All the commit are avialable in the github link [7] : https://gitlab.com/PranabNandy/ns-3-dev/-/commits/PI

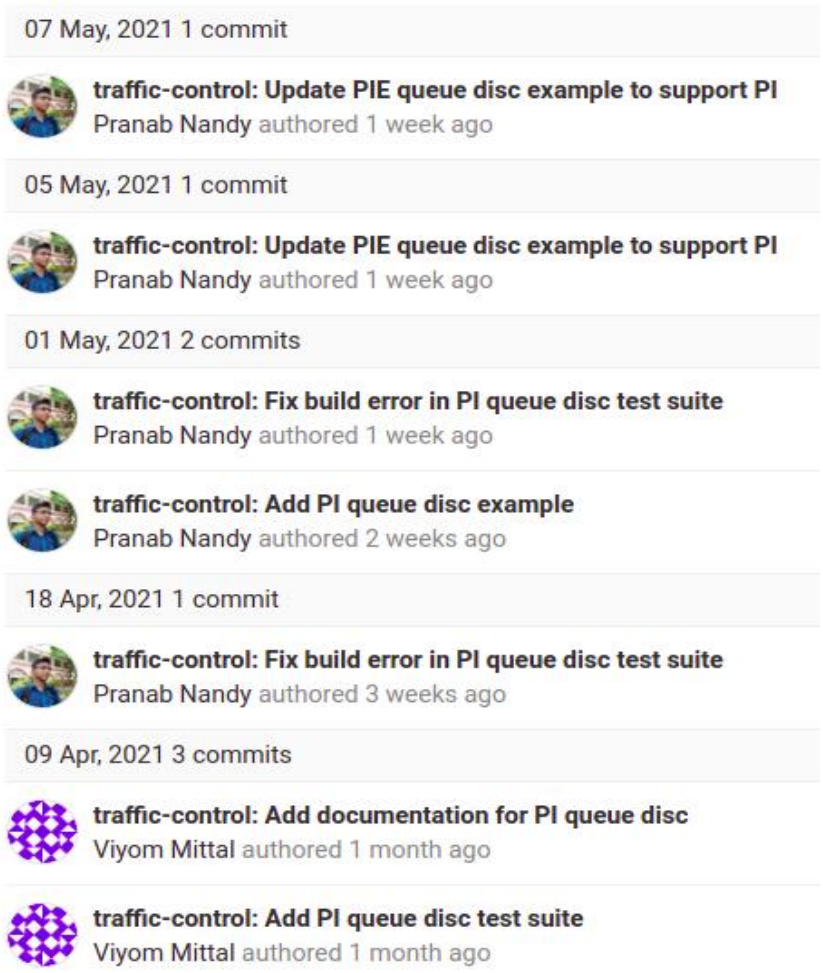


Figure 3.5: Commits on PI branch [7]

Chapter 4

Model Evaluation

We have designed unit tests for verifying the implementation of PI model in ns-3. These tests are mandatory to ensure the correctness of new models before they can be merged into ns-3-dev. Additionally, we validate the correctness of the proposed model by testing both algorithms under various network conditions. Then We compare their performance against the expected behiviour which we obtained from other model. The performance metrics used for comparison are throughput, congesion window and queue length.

To evaluate the model performance we created the dumbbell topology with 2 senders, 2 Recievers and 2 Rounter. We have used various RTT for different senarios for 2 different flows S1 to R1 and S2 to R2 respectively. The fig 4.1 depicts the dumbbell topology for the scenario 1.

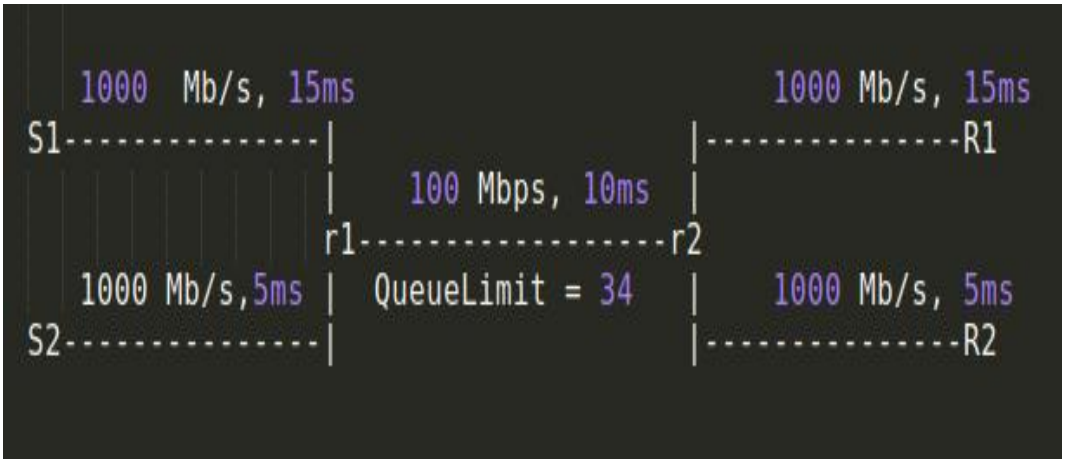
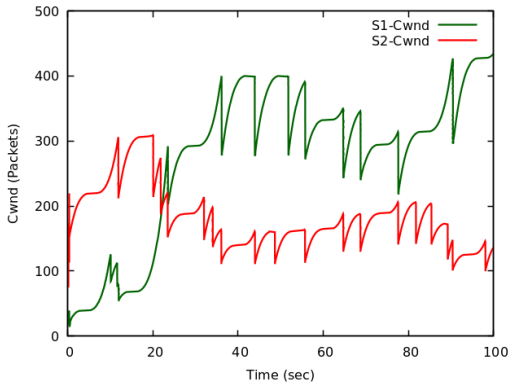
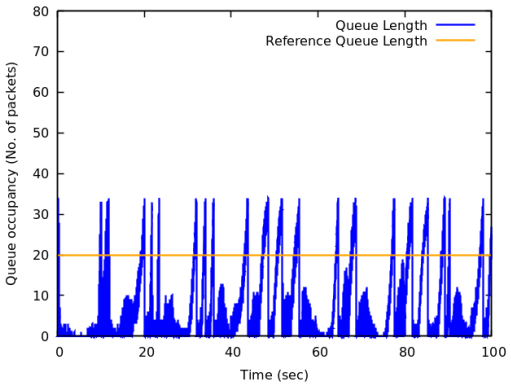


Figure 4.1: Dumbbell Topology

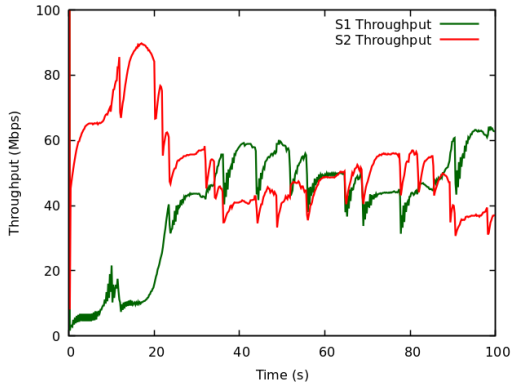
4.1 Scenario 1 : Different RTTs



(a) Congestion Window vs Time



(b) Queue Size vs Time



(c) Throughput vs Time

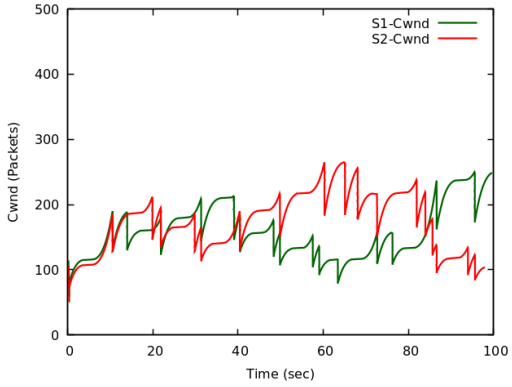
Figure 4.2: Result on Different RTT scenario

Table 4.1: Scenario 1 configuration

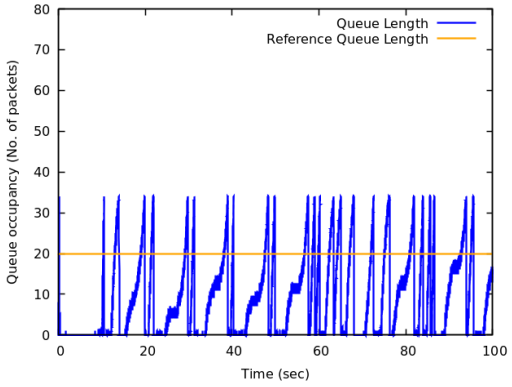
Parameter	Value
Topology	Dumbbell
Mean packet size	1448 B
TCP	TcpNewReno
Queue Size	34 packets
RTT between S1 and R1	80ms
RTT between S2 and R2	40ms
α	0.00001676898811
β	0.0000167434834
W	170
QueueRef	20 packets
Simulation duration	100s

Here the fairness of Cwnd of two senders are highly impacted due to using different RTTs for 2 senders. Initailly both folws do not comverge upto 25 secs. Cwnd of S2 is more initailly that is why S2 has more throughput in initail 25 secs. After that though S1 has higher Cwnd but they have almost same thoughput.

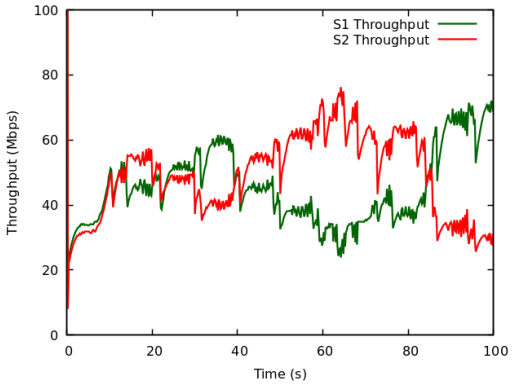
4.2 Scenario 2 : Same RTTs



(a) Congestion Window vs Time



(b) Queue Size vs Time



(c) Throughput vs Time

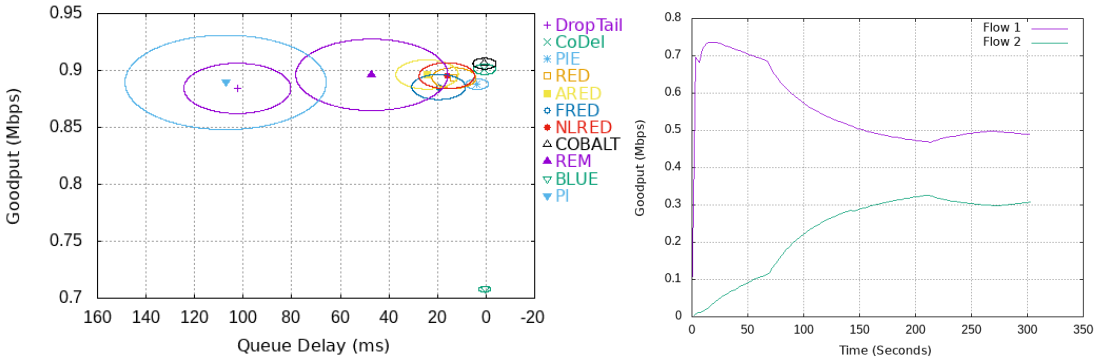
Figure 4.3: Result on Same RTT scenario

Here the fairness of Cwnd of two senders are very good as compared to different RTTs for 2 senders. Whenever S2 has larger Cwnd as compared to S1, its throughput is higher than S1 sender in those times.

Table 4.2: Scenario 2 configuration

Parameter	Value
Topology	Dumbbell
Mean packet size	1448 B
TCP	TcpNewReno
Queue Size	34 packets
RTT between S1 and R1	40ms
RTT between S2 and R2	40ms
α	0.00001676898811
β	0.0000167434834
W	170
QueueRef	20 packets
Simulation duration	100s

4.3 Scenario 3 : Mild Congestion in AQM Suite

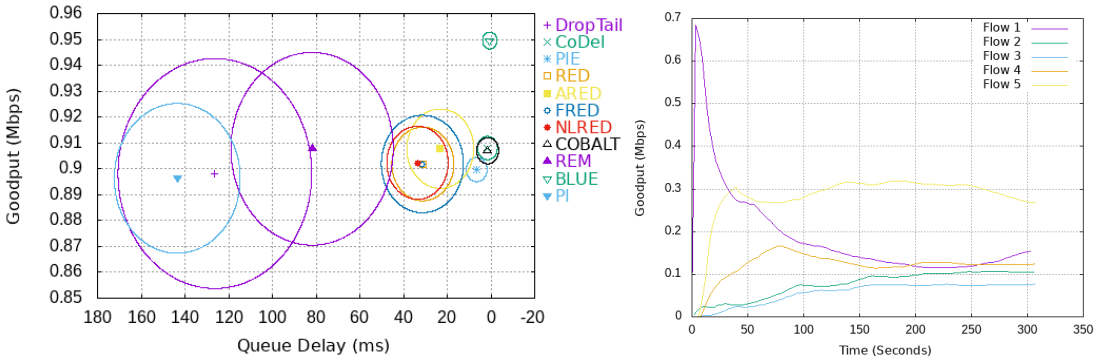


(a) Goodput vs Queue delay with other AQM model (b) Goodput vs Time

Figure 4.4: Result on Mild Congestion in AQM Suite

Here 2 flows are being used to create the mild congestion environment. The flow 1 starts with good goodput then its goodput is degraded over some time. Whereas the flow 2 starts with low goodput and its goodout improves over time. After cetain time both flows got saturated when AQM can able maintain the desired queue length. And also the variation of queue delay of PI queue disc is less considering the goodput it is providing over queue delay.

4.4 Scenario 4 : Medium Congestion in AQM Suite

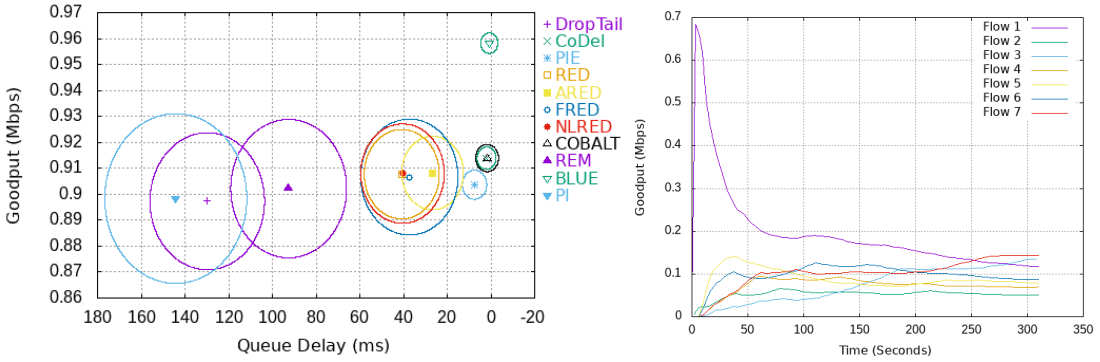


(a) Goodput vs Queue delay with other AQM model (b) Goodput vs Time

Figure 4.5: Result on Medium Congestion in AQM Suite

Here 5 flows are being used to create the Mediun congestion environment. The flow 1 starts with good goodput then its goodput is degraded over some time. Whereas the flow 5 starts with low goodput and its goodout improves over time. Other 3 flows can not able to increase or decrease its good put significantly over the time. To some extent we can say others 3 flows improve their goodput. And also the variation of queue delay of PI queue disc is medium considering the goodput it is providing over queue delay as compared it with other AQM algorithm.

4.5 Scenario 5 : Heavy Congestion in AQM Suite



(a) Goodput vs Queue delay with other AQM model (b) Goodput vs Time

Figure 4.6: Result on Heavy Congestion in AQM Suite

Here 7 flows are being used to create the Heavy congestion environment. The flow 1 starts with good goodput then its goodput is degraded over some time. Other 6 flows can not able to increase or decrease its good put significantly over the time. To some extent we can say others 6 flows improve their goodput. After certain time no

of the flows can able to improve its good put due to heavy congestion. And also the variation of queue delay of PI queue disc is very high as compared it with other AQM algorithm.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this project, we started with the requirement of queue management algorithm in network to control the congestion. Then we see the drawback of PQM and understand the need of AQM. Then we discussed why we have considered PI queue disc for our project. Its code is already implemented by researchers. We have upgraded the existing code so that it can fit with recent c++ comipler of ns-3. Then we evaluate its perfromance through the popular dumbbell topology. After that we used AQM test suite to perform a through analysis over different network congestion. We are planning to merge it with the mainline of ns-3.

5.2 Future Work

At the time of evaluating the PI model, ECN is currently not supported in the main code of PI controller. ECN can be used in order to distinguish between the Classic and Scalable TCP flows. Therefore, we have to implement the ECN bit feature in the dropEarly() function along with drop probability decision making. Here we are using the constant parameter like α, β in probability calculation which might work for some network condition but it may not work other network condition. For overcoming that problem we might have to go with Self Tuning feature along with PI controller . The Self tuning feature [6] enable the PI controller to adjust the controller according to newtork situation.

Bibliography

[1] Vivek Jain, Viyom Mittal, Shravya K. S., Mohit P. Tahiliani , Wireless Information Networking Group (WiNG), Department of Computer Science and Engineering NITK Surathkal, Mangalore, Karnataka 575025, India et al, BLUE and PI :" Implementation and validation of BLUE and PI queue disciplines in ns-3",Simulation Modelling Practice and Theory 84 (2018) 19–37.

[2] T. A. N. Nguyen, T. Henderson, D. Taht. Understanding bufferbloat through simulations in ns-3, 2014, (https://www.nsnam.org/wiki/ GSOC2014Bufferbloat).

[3] B. Braden , D. Clark , J. Crowcroft , B. Davie , S. Deering , D. Estrin , S. Floyd , V. Jacobson , G. Minshall , C. Partridge , et al. , Recommendations on Queue Management and Congestion Avoidance in the Internet, Technical Report, IETF, 1998 .

[4] PID controller from wiki pedia : `https://en.wikipedia.org/wiki/PID_controller`

[5] C.V. Hollot , V. Misra , D. Towsley , W.-B. Gong , "On designing improved controllers for AQM routers supporting TCP flows ", in: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), 3, IEEE, 2001, pp. 1726–1734 .

[6] Honggang Zhang, C. V. Hollot, Don Towsley ,Department of Computer Science ,University of Massachusetts at Amherst and Vishal Misra ,Department of Computer Science, Columbia University, USA. et al. Self Tuning PI :"A Self-Tuning Structure for Adaptation in TCP/AQM Networks.",IEEE, 2013, pp. 148–155 .

[7] My Gitlab Repository :
`https://gitlab.com/PranabNandy/ns-3-dev/-/commits/PI`

[8] S. Murali , M.P. Tahiliani , et al. , "Implementation and Evaluation of Proportional Integral Controller Enhanced (PIE) Algorithm in ns-3" : Proceedings of the Workshop on ns-3, ACM, 2016, pp. 9–16 .