Congestion Control in TCP:BBR

B.Tech Major Technical Project Report

to be submitted by

Divyasheel Kumar

b19081

for the partial fulfillment of the degree

of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

KAMAND-175075, INDIA

November, 2022

Contents

1 Introduction **2**

1.1 The problem with loss-based congestion control. 2

1.2 What is TCP:BBR? 3

1.3 Motivation for the project 4

2 Approach **5**

2.1 The problem statement 5

2.2 Method 5

2.3 System 6

3 Simulations **7**

3.1 Tools Used 7

3.2 Process 8

3.3 Network Topology used 8

3.4 Important features added into the Simulation 8

3.5 Focus of the simulations 9

3.6 Results 9

3.7 Conclusions 11

4 Modeling **13**

4.1 Why do we need modeling ? 13

4.2 The 2 Phase Model 13

4.3 Assumptions made by the 2 phase model 15

4.4 Progressing from the 2 phase model (FUTURE WORK) 15

4.5 Conclusion 16

List of Tables

List of Figures

1.1 throughput comparison showcased by google 3

2.1 n-dumbell design 6

Chapter 1

Introduction

1.1 The problem with loss-based congestion control.

In loss based congestion control mechanisms, congestion control is triggered by packet loss. One of the reasons for packet loss is a bottleneck queue getting completely filled. Once these queues are filled to the brink, any new packets that come in are dropped. This packet drop is considered as a sign of a congested bottleneck. But in order to detect this congestion, the queue must be first filled to the brink. Otherwise there would be no packet loss. Therefore, this mechanism forces the queue to be filled before acting on decreasing the congestion.

With the advancement in hardware resources, we can afford large buffer sizes in these bottleneck link queues. Now, these large queues also also eventually get filled up if there is congestion. In fact, now the waiting time (queuing delay) is a lot more. Therefore, the loss based congestion control mechanisms will not detect and act on the congestion until these large queues are filled. But by then it's already too late. These large queues are already adding a lot of latency to the system.

This is known as the Bufferbloat problem and is the cause of increased latencies in many networks.

Another problem experienced by loss-based congestion control mechanisms is that

of being too reactive to packet loss from shallow buffers. These shallow buffers therefore cause the congestion control mechanism to kick in and substantially reduce the throughput.

Therefore, loss-based congestion control can lead to increased latency and reduced throughput.

1.2 What is TCP:BBR?

BBR stands for - "Bottleneck Bandwidth and Round-trip propagation time". It is a fairly new congestion control algorithm developed by Google.

BBR is a model based congestion control mechanism. Model based here means that it tries to model the network keeping in mind 2 important parameters - bottleneck bandwidth and round trip propagation time. Using these 2 parameters it tries to control the packets in flight released by the sender. Therefore instead of waiting for packet loss, it continuously tries to study the network and figure out the optimal value of packets in flight.

This approach is fundamentally different than the traditional congestion control algorithms because it does not wait for signals (packet loss or increased delay) for a "full pipe" but tries to figure out how "full" this pipe should be. This amount is considered using the Bandwidth Delay Product.

Using this approach Google claims to achieved groundbreaking results that offer as much 2700x higher throughput and 25x times lower queuing delays [1].

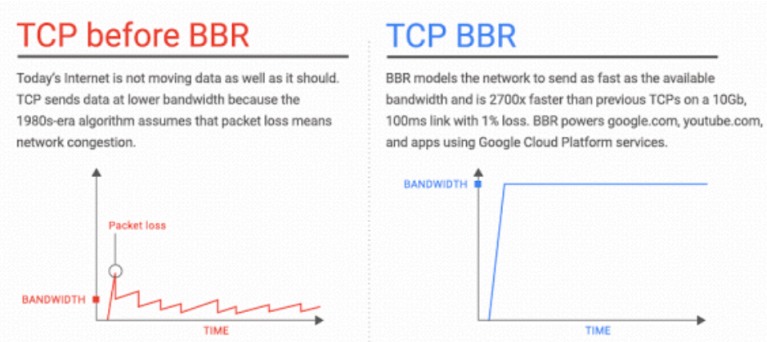


Fig. 1.1: throughput comparison showcased by google

1.3 Motivation for the project

Google is a household brand. It owns a lot of services that have become essential to our digital lives. Considering that TCP:BBR is Google's own project, this project is bound to be adapted into these very services and eventually make its way to the internet.

Although Google claims the algorithm to be ready considering the huge leap in performance that they reported, researchers have been skeptical of these claims. Some of the researchers have also raised issues regarding fairness and the non-conservative ambitious approach used by BBR [2].

One of the reasons for this skepticism is the fundamentally different nature of BBR. It is not simply an addon or improvement to the previously used age old TCP algorithms, but was made from scratch. The previously used TCP variants, although not completely efficient, but are well looked into ever since their inception. Their fundamentals are well researched and shortcomings known well in advance. But this is not the case with BBR as it is a fairly new algorithm with very little external research to support it.

The internet is very complicated as there is huge overlap between different flows and bottlenecks. BBR's approach tries to model this very internet considering individual flows and their bottlenecks. But it does not provide any special attention to overlapping configuration and hence dependent nature of these flows. Things like fairness and sharing are also not discussed.

Considering that BBR is backed by Google, it has a high chance of making its way to the internet. It is already being used by services like Dropbox and Spotify. Therefore, it is very important that we test out its impact and performance over the internet before it reaches that level. This will help us prepare and suggest modifications to the algorithm.

Therefore, the motivation of this project is to test out whether BBR's model based approach will hold for the internet, what kind of performance should we expect and what are the factors that influence this performance.

Chapter 2

Approach

2.1 The problem statement

To test out the performance and impact of TCP:BBR's model based approach on n overlapping flows with shared bottlenecks and how this system evolves as n tends to infinity.

In simpler words, we want to study how BBR responds to the complicated aspects of the internet by using a toned down model that involves n different flows. We will focus on how these flows interact with each other and how it affects the queues at these shared bottlenecks.

Once we have an apt model in place, we need can extend these n flows to infinity to gain understanding of a system similar to the internet.

2.2 Method

Now that we have the problem statement in place. We can go about solving this problem using 2 methods. These methods are-

- Using Simulations - In this method we run simulations in topologies and configurations that are relevant in real world scenarios, and help us gain useful insight into factors that are responsible for producing a favorable outcome.

- Using Mathematical Modelling - In this method we use differential mathematics to create models of the system whole behavior we are trying to understand. These models also allows us to build relations between different parameters of the system and solve qualitative questions.

Both these methods are essential to produce a working model that will then be used to analyze TCP:BBR's performance and impact.

2.3 System

Considering that the aim of the project is to predict TCP:BBR's performance and impact on the real world internet, we need to configure a system that is a toned down version but still focuses on certain important aspects like sharing and synchronizing. This system will act as our playground and help us test out the model.

To design such a system we must first consider the important non-negotiable aspects. These aspects are bottleneck sharing, flow synchronization, queue utilization and throughput.

Keeping in mind all the aspects, the design that we will be using is shown below.

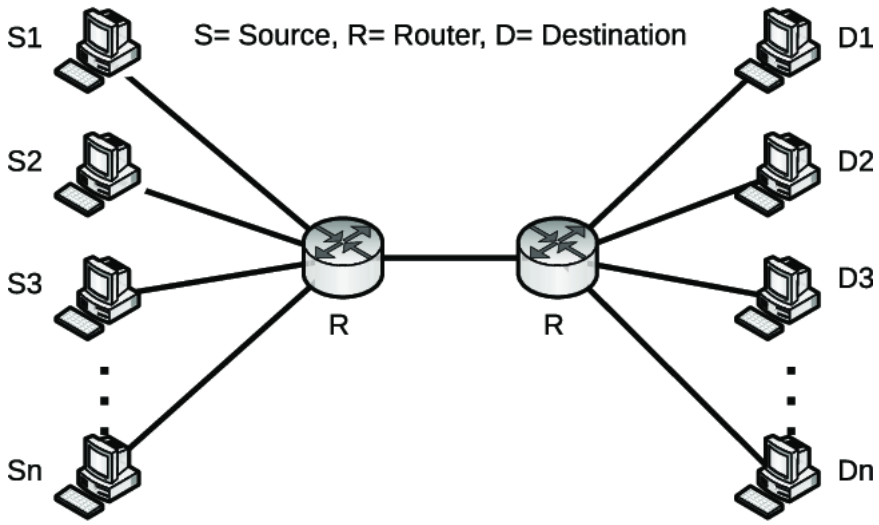


Fig. 2.1: n-dumbell design

Chapter 3

Simulations

3.1 Tools Used

We used the following tools/frameworks to perform our simulations:

- NS2 and NS3 - For testing the performance attributes of different TCP algorithms in variable network topologies and constraints like bandwidth and delay. It is also responsible for generating time series data for each simulation.
- Python - For automating the entire process of
  - Running different sets of multi-threaded NS3 simulations generating different combinations of inputs.
  - Parsing the time-series data of each simulation from files generated by NS3.
  - Using this parsed information to plot graphs for different parameters of performance like "cwnd_vs_time", "dropped_packets", "link_utilization", "qsize_vs_time" etc.
- Jupyter Notebook - For storing and presenting the plots generated for each simulation run.

3.2 Process

The simulation are performed using the tools mentioned in the previous section in the following manner order

- Run the main ipynb script with inputs like Bottleneck bandwidth, Delay etc.
- The ipynb script runs different combinations of multi-threaded ns3 simulations
- The ns3 simulations create time-series data for each simulation
- The ipynb script is again used to parse this time-series data and plot graphs for useful insights

3.3 Network Topology used

The network topology used is similar to the design in Fig.2.1.

It consists of n nodes for source and n nodes for destination. We add 2 additional nodes (r1 and r2) to create the bottleneck link. The n source nodes are connected to r1 using n p2p links. The node r1 is connected to r2 using the same p2p type link. The node r2 is then further connected to n sinks using n p2p links. The bottleneck link r1-r2 has a queue in place using DropTail policy.

Here, n is an input parameter and can be changed for each simulation along with Bottleneck bandwidth and Delay.

3.4 Important features added into the Simulation

- **Start time for each flow is randomized** - There are N flows in the N dumbbell topology but if setting all their starting time to be same is not very realistic , therefore we have programmed their starting time to follow the poisson distribution.
- **Optimized writes to trace files** - When dealing with the congestion window, we have N sources that write to N files simultaneously to record the change in

their window size. Now this window size changes a lot per second. This is a lot of IO for the operating system if the value of N is large. This was optimized by disabling the trace after every windows size changes for all N. Therefore a custom function that writes the current value to a file after a fixed interval of time was coded.

- **Multi-threaded simulations** - NS3 doesn't allow us to run simulations in parallel but considering that we need the results of a lot simulations to understand what change in parameters is causing the effect, we need to run a lot of simulations. Therefore, to allow multiple parallel runs of NS3 itself, we used python threads and locks to decrease the running time from 30 mins. down to 3-4 mins.

3.5 Focus of the simulations

Using these simulations we want to gather information about the performance in terms of Throughput and Latency, as discussed in 1.1. But both these metrics are not readily available in NS3 but need to be deduced from other parameters. The readily available parameter chosen for this were Link utilization and Queue utilization respectively.

Link Utilization here refers to the percentage of the bottleneck link being used. Because the bottleneck link will limit after flow's throughput, the total amount of throughput generated by this system is the same as the utilization of this bottleneck link.

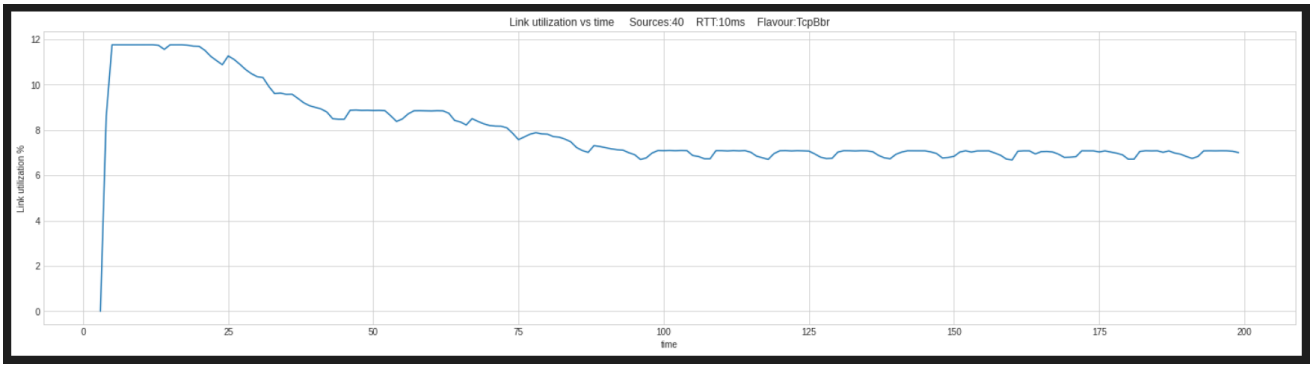
Queue Utilization is used to deduce the latency because if the amount of queue being utilized grows, the waiting time (queuing delay) caused by this bottleneck link will increase. Therefore higher the Queue Utilization, more is the latency.

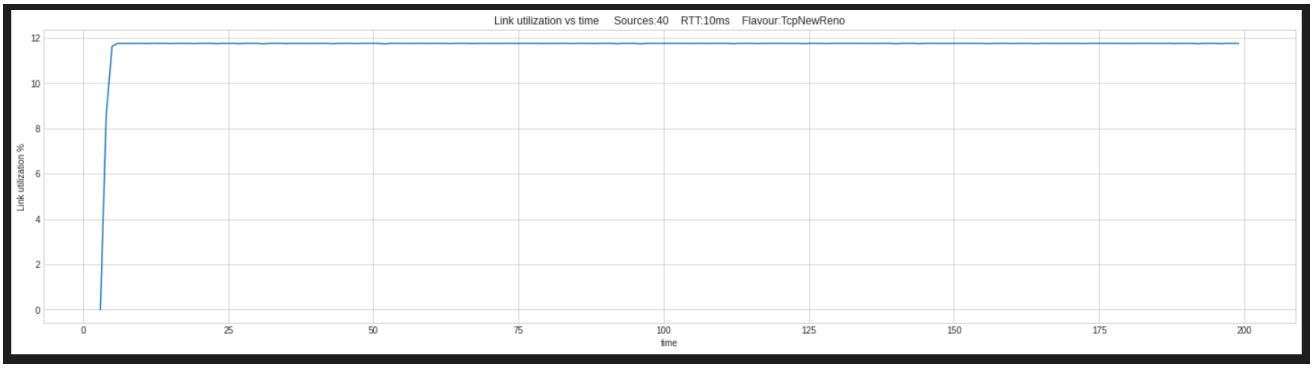
3.6 Results

After setting up the simulations, we ran the simulations on 2 different TCP sockets for comparison: BBR and NewReno. NewReno is also relatively updated TCP flavor but relies on the same loss-based mechanism. We compared BBR's performance with

NewReno on the basis of link utilization, queue utilization (as per 3.5), along with packets dropped and the congestion window size. The results were as follows:

- Link utilization (throughput): Here, we can observe that TCP New Reno offers

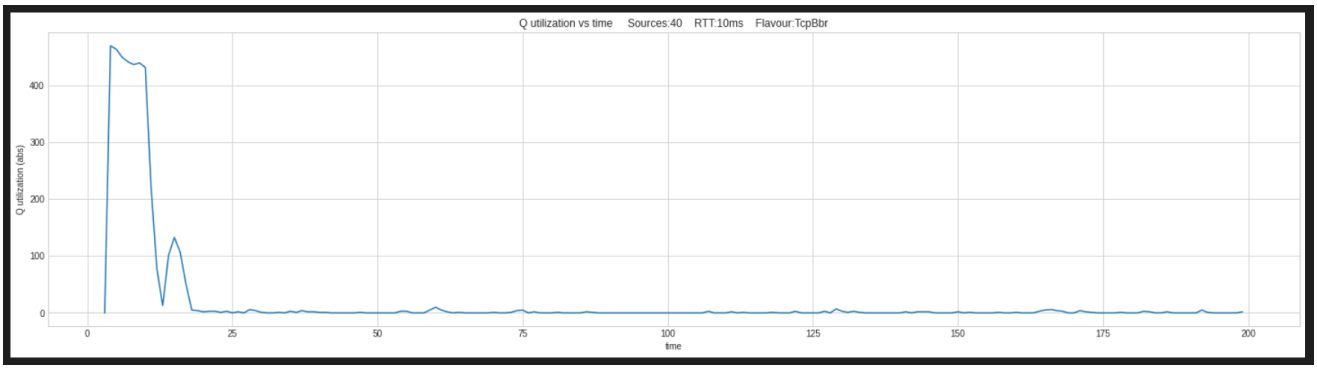


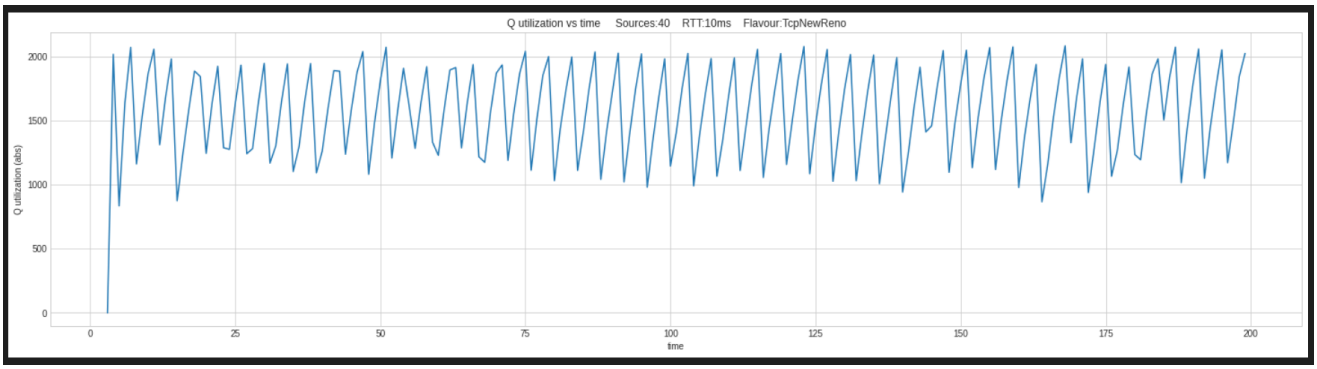


  better link utilization then TCP BBR. This goes against claims made by Google. Now the cause of this failure for BBR is something that can be investigated.

- Queue Utilization (latency): We can observe that BBR's queue remains much less filled than NewReno. This means that BBR offers much less latency due to queueing delays than New Reno. Here the claim made by Google stands true. Also, this empty queue can be a reason for lesser link utilization for BBR in 3.6 because the link has to wait for the queue to be filled with packets to be put on the link. Whereas for New Reno this queue is filled with ready to serve packets for the link, therefore more link utilization.

- Packet Drop and Congestion Window in BBR and NewReno: Considering that BBR does not wait for packet loss to act on congestion, we can see that the size





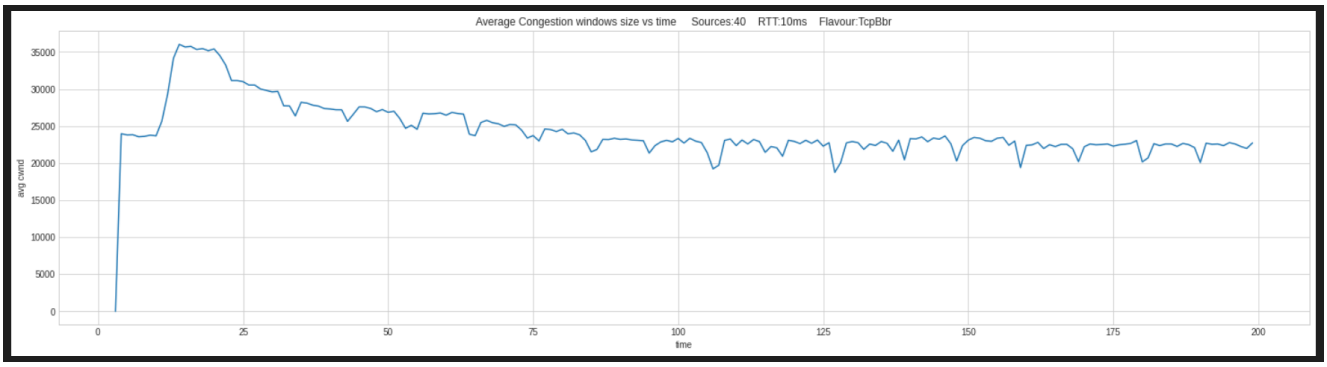
of the congestion window is modulated with any packet loss. But in the case of New Reno, the congestion window only starts changing after packet loss occurs.

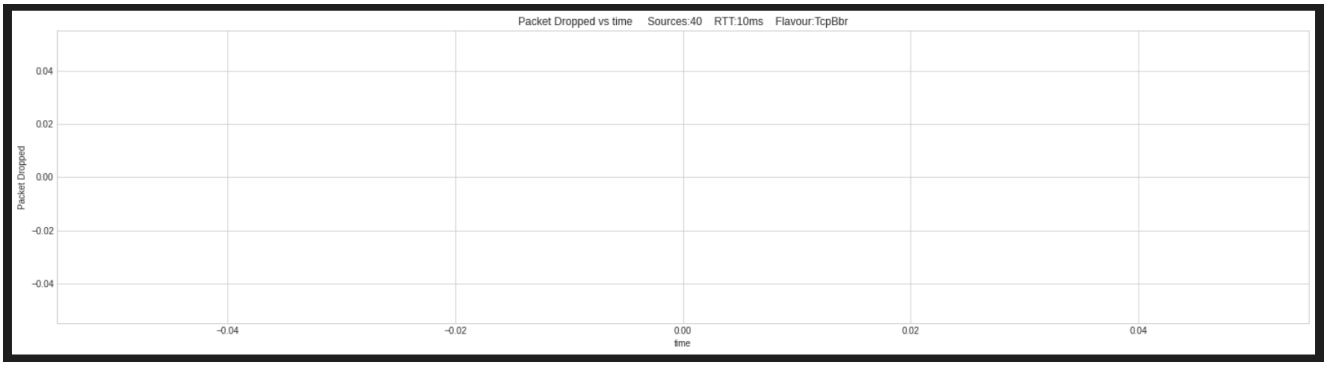
3.7 Conclusions

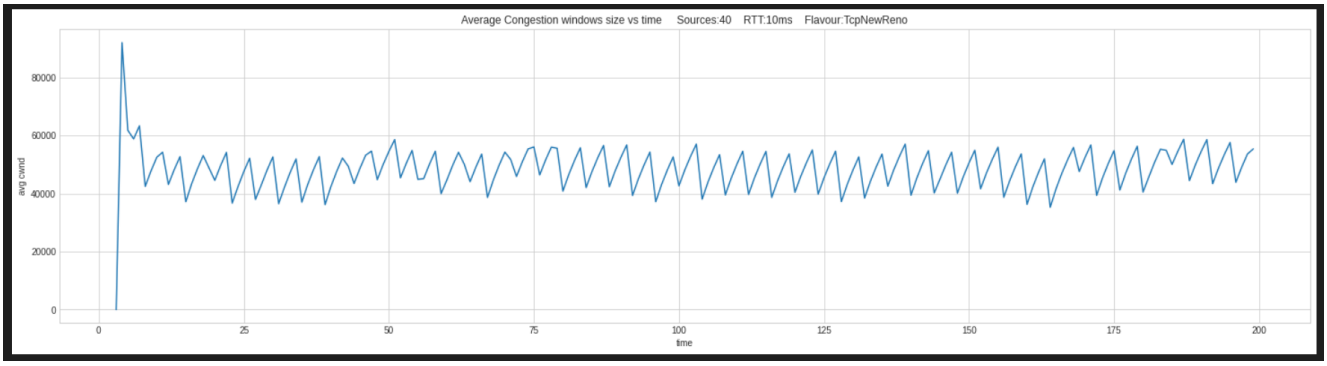
The following conclusions can be made on the basis of the simulations performed.

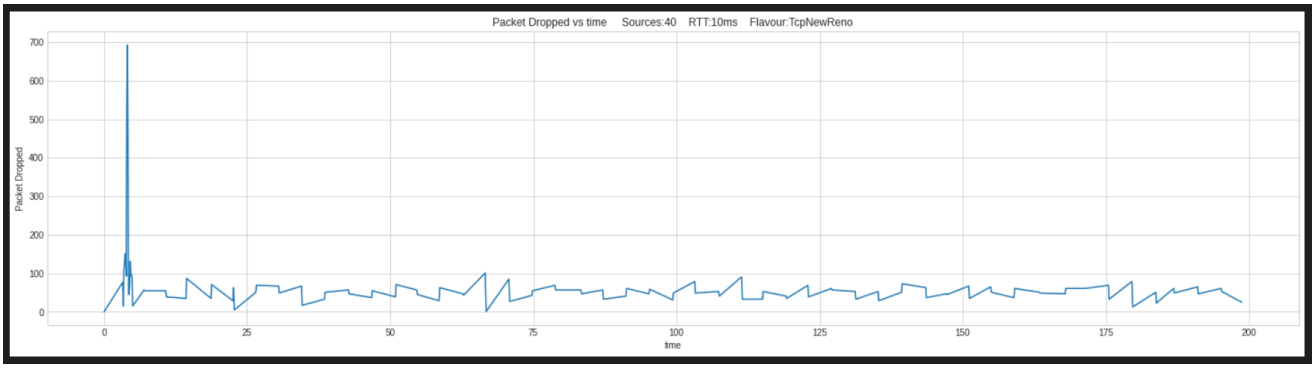
- BBR is good at reducing queuing delays: We can observe that there is stark difference between the queue utilization for BBR and NewReno. BBR only fills its queue till about 400 packets and then immediately drains it to keep it at a very low level throughout.

- BBR's throughput might suffer due to its unutilized queue buffers: We observed that the throughput in terms of the bottleneck link Utilization was lesser in case of BBR. The reason for this can be the empty queue buffer that BBR produces to decrease latency.


Average Congestion windows size vs time Sources:40 RTT:10ms Flavour:TcpBbr
avg cwnd
time


Packet Dropped vs time Sources:40 RTT:10ms Flavour:TcpBbr
Packet Dropped
time


Average Congestion windows size vs time Sources:40 RTT:10ms Flavour:TcpNewReno
avg cwnd
time


Packet Dropped vs time Sources:40 RTT:10ms Flavour:TcpNewReno
Packet Dropped
time

Chapter 4

Modeling

4.1 Why do we need modeling ?

We need modeling to ensure that our simulation results are backed by mathematical proofs. Mathematical modeling is also better at understanding how the system will behave when it is pushed into abstract regions like infinity. This model can also be used to understand the relation between different parameters of the system. Modeling is also much cheaper in terms of resources and doesn't require any computation. Therefore, modeling can be used to guide our simulations without wasting computation.Simulations are based on causality but mathematical models are based on input and output parameters along with significance to their weighted relations. It is much easier to understand a system using these relations than using multiple hits and trials in a simulation.

4.2 The 2 Phase Model

To develop a model we must first start with a basic understanding of how the input and output parameters relate to each other. To understand this we started with a simple 2 phase model [2] where the relation is derived for a single flow.

We know that BBR focuses on "packets in flight" in order to reduce congestion but according to this model it does so in 2 phases, namely-

- Pace controlled phase
- Congestion window controlled phase

This model also introduces an M.I.M.D. factor that is used by both these phases for modulation. This factor takes as input the RTT of a flow and the queuing delay caused by the bottleneck buffer. Using this factor the modulation can be described as:

Pace controlled phase - This phase is activated when BBR is either accurately estimating or underestimating the bottleneck bandwidth. Here, the packets in flight are limited by pacing. This is done by changing the estimated bottleneck bandwidth which in turn changes the packets in flight. The relation for estimated bandwidth is -

$$C_i(t) = \frac{1.25T_i}{T_i + D(t)} \times C_i(t - \Delta_i)$$

Where, $C_i(t)$ is the estimated bandwidth at time t, T_i is the RTT of the flow and $D(t)$ is the queuing delay at time t. Also i is the ith flow being referred to.

Although the modulation is done through pacing here, the congestion window size also changes.

Congestion window controlled phase - This phase is activated when BBR is overestimating the bottleneck bandwidth. Here, the packets in flight are limited by the size of the congestion window. The relation for the change in the congestion window is -

$$W_i(t) = \frac{2T_i}{T_i + D(t)^{MIN}} \times W_i(t - \Delta_i)$$

Where $C_i(t)$ is the estimated bandwidth at time t, T_i is the RTT of the flow and $D(t)$ is the queuing delay at time t. Also i is the ith flow being referred to.

Therefore the 2 phase model can be summarized using the M.I.M.D. factor as:

$$MIMD_i = \begin{cases} 1.25T_i/(T_i + D(t)) & \text{in pacing controlled phase} \\ 2T_i/(T_i + D(t)^{MIN}) & \text{in Cwnd controlled phase} \end{cases}$$

4.3 Assumptions made by the 2 phase model

The 2 phase model although instrumental in creating a base for understanding how BBR works, makes the following assumptions-

- Multiple flows share the same bottleneck link - This is not true for the internet as 2 different flows can easily have 2 different bottlenecks. It is possible that there is another bottleneck buffer outside the overlapping part of the flows.
- The flows all start at the same time - The model further goes on to assume that the flows are competing for a shared buffer which is initially empty. This can only happen if the flows all start at the same time which is a very unrealistic assumption.
- Tendency to stay in the Congestion Window controlled phase - The paper later goes on to make an assumption that the the system will most probably stay in the congestion window controlled phase. This assumption is made by claiming that when a bottleneck is shared by multiple flows, the bandwidth is mostly overestimated.

4.4 Progressing from the 2 phase model (FUTURE WORK)

Although, the 2 phase model helps us understand how modulation happens in BBR, the assumptions like the independent non-synchronous nature of 2 different flows and the condition of having the same shared bottleneck can be worked on.

We can also further apply this model to our N-dumbbell design [2.1] and then extend this N to infinity to gain inference on how the algorithm would perform in the real world.

We can also consider generalizing this model further for considering flows with different TCP sockets like NewReno, Compound etc. This would help us understand how the algorithm performs flows that are using the loss based mechanism. This is

important because BBR aims at keeping the queue empty but the loss based algorithm requires the queue to overflow in order to detect congestion.

4.5 Conclusion

There is a lot work still left to be done on the modeling end but our knowledge of how BBR functions from the 2 phase model is a good starting point for progressing on to a more general and realistic scenario of infinite flows that overlap all over the internet.

There are also some assumptions made by the 2 Phase model and we can use our simulations to test if they hold true or not and why?

The first thing that needs to be changed in this model is the independent nature of these overlapping flows. We need to add some kind of a shareable constraint on these flows that makes this construct more realistic.

Most of the work ahead will be focused on modelling with the aid of simulations.

Bibliography

[1] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *ACM Queue*, vol. 14, September-October, pp. 20 – 53, 2016. [Online]. Available: http://queue.acm.org/detail.cfm?id=3022184

[2] M. Zhang, M. Mezzavilla, S. Rangan, and S. Panwar, "Improving google's bbr for reduced latency and increased fairness," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 1–6.