Study of Congestion Control Algorithms in Packet-Switched Networks

A Project Report Submitted by

Somit Gond

in partial fulfilment of the requirements for the award of the degree of

B. Tech

Indian Institute of Technology Mandi

Computer Science

April, 2025

Declaration

I hereby declare that the work presented in this Project Report titled Study of Congestion Control Algorithms in Packet-Switched Networks submitted to the Indian Institute of Technology Mandi in partial fulfilment of the requirements for the award of the degree of B. Tech, is a bonafide record of the research work carried out under the supervision of Prof. Padmanabhan Rajan, Prof. Sreelakshmi. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Signature

Somit Gond

B21138

Certificate

This is to certify that the Project Report titled Study of Congestion Control Algorithms in Packet-Switched Networks Report, submitted by Somit Gond(B21138) to the Indian Institute of Technology Mandi for the award of the degree of B. Tech, is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Prof. Padmanabhan Rajan, Prof. Sreelakshmi

Abstract

Today's world can't be imagined without internet. We connect our devices to internet throught routers. Router transfer packets(data) from one link to another, forming internet infrastructure. Everyday internet traffic keeps rising, but internet's infrastructure doesn't scale as fast. This creates huge load on existing infrastructure, leading to scenarios of network congestion. A Network congestion forms at a router when rate of incoming packets is greater than bandwidth of outgoing link. Routers have finite buffer and they drop packet when it gets full. Various methods have been developed over the years to reduce network congestion. Different flavours of TCP have been developed to efficiently handle congestion. TCP takes multiple packet loss as a sign of network congestion so it starts to slow down its sending rates. Routers leverage this behaviour of TCP in their favour they can drop packets early to notify clients that congestion is happening or will happen. Router can do this by managing its queue size. Threshold based queueing policy is one of the methods where router starts to drop packets when queue size exceeds a threshold. Global synchronization is one of the issues arising due to this policy in which either all clients back off simultaneously or start sending packets simultaneously, resulting in underutilization of bandwidth of link. It is essential to set threshold such that global synchronization doesn't happen. We want to use data driven techniques to compute the queue threshold (q_{th}) for threshold based queueing policy.

Contents

List of Figures

Study of Congestion Control Algorithms in Packet-Switched Networks

1 Introduction and background

Today's world runs on internet. The demand for internet is increasing every second but infrastructure supporing internet is not scaling as fast.

Congestion happens at a router when the rate of incoming packets is greater than the bandwidth of outgoing link. Routers have a finite buffer so that it can store the packets that can't be sent immediately, sending when bandwidth is empty or not full, thus ensuring full utilization of bandwidth. But what if the buffer is full at the router, it can't store more packets, so it drop packets. TCP uses this implicit signal of dropping packets as sign of congestion. We can tune this packet dropping behaviour to notify clients of congestion early. For that we have various methods called Active Queue Management policies. The router can drop packets when its buffer gets full but it creates a situation of bufferbloat i.e its buffer is never empty. To ensure that this situation doesn't occur Random Early Detection (RED) was developed. In RED, router starts to drop packets with some probability, the probability depends on the moving average of queue size.

Some of Active Queue Management (AQM) algorithms used in today's routers :

1. Threshold Based Queueing: If queue size is beyond a threshold, the router starts to drop packets.
2. Random Early Detection: Router starts to drop packets with probability dependent on queue size i.e when queue is empty the probability is 0 and when queue is full the probablity is 1 and in between it calculates the probability and drops packets.

Our goal in this project is to how to calculate parameters for threshold based queuing mechanism.

2 Literature survey

Transmission Control Protocol (TCP) is a reliable, in-order, byte-stream transport layer protocol [3]. TCP reliability constitutes of error correction mechanism, detecting packet loss and correction via retransmission [3].
TCP uses various congestion control mechanisms [2] :

- Slow Start: At start of a connection, trasmission rate (Congestion window (cwnd) size) is increased exponentially until a threshold called slow start threshold (ssthresh) is reached or congestion is detected.
- Congestion Avoidance: After threshold is reached, TCP increases congestion window linearly until bandwith limit is reached.
- Fast Retransmit and Fast Recovery: When packet loss is detected TCP retransmits the package and enter fast recovery to recover from the loss.

Some notable improvements include:

- Reno [2]: Improving fast Retransmit and fast recovery aspect of TCP.
- NewReno [5]: Improvement over Reno, improves retransmission during fast-recovery phase of Reno.
- Cubic [8]: Uses cubic function instead of linear function for congestion window increase. It is designed for high speed and long distance network.

Routers has some finite buffer in which it stores the packets that it can't send immediately. Generally routers drop incoming packets when its buffer is full. But using, Active Queue Management (AQM) techniques it tries to manage buffer, dropping packets before buffer gets full or using a some criteria for dropping packets[1]. Random Early Detection (RED) is one such AQM where, when the average queue size exceeds a preset threshold the router drops incoming packet with a certain probability, where exact probability is a function of the average queue size [4]. RED is highly dependent on initial parameter setting and is not well performant with bursty traffic. Various changes have been suggested to RED such as Exponential RED [6], PIE [7] etc. While in Threshold based queueing policy router drops incoming packet when queue size exceeds a certain threshold. Some disadvantages of threshold based queueing policy are burst losses i.e all packets are dropped suddenly, flow syncronization i.e all flows are either backing off simultaneously or sending packets simultaneously.

3 Problem definition and Objective

TCP sends multipule packets at the same time. The set of packets is called sending window or cwnd. TCP changes the size of cwnd during different states it is in like slow start, congestion avoidance. In congestion avoidance phase of TCP it follows additive increase and multiplicative decrease to change cwnd size.

Additive increase: $w_{t+1} = w_t + \alpha w_t^{k-1}$

Multiplicative decrease: $w_{t+1} = (1 - \beta)w_t$

where $\alpha > 0, \beta < 1$

The value of q_{th} can be calculated as:

$$\beta w^* \left(\frac{w^*}{C\tau}\right)^{q_{th}} q_{th} = \pi/2$$

where q_{th} is queueing threshold, C is link bandwidth, τ is average round trip time, β is AIMD parameter (generally 0.5) and w^* is average window length.

Now, how to calculate the parameter q_{th} effectively.

To calculate of q_{th} we require values of β and w^*.

So, the objective of our project is to:

1. Simulate sample topologies in ns-3 and Generate Data
2. Calculate β value and w^* from generated data, then calculating q_{th}
3. Implement this whole process of calculating q_{th} then setting it as threshold, in ns-3

4 Methodology

The methodology we took to achieve the desired objective:

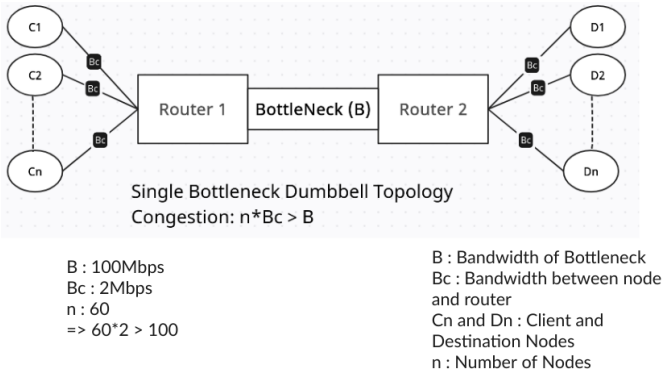

Figure 4.1: Topology used for Simulation

1. Literature Review: We have done extensive literature review on our topic. By doing this we found at what level current state of the art is. We also learnt about different approaches explored to counter network congestion.

2. ns-3: ns-3 is a network simulator program. It is open-source so anybody can contribute in it. It is used to simulate different network scienarios. It uses C++ Programming languague at is base. The topology can be build using C++ code. It is a network simulator so all the pieces required for simulating a network scienario exist in it. We learnt how to use ns-3 to simulate example topologies.

3. Topology Generation: 4.1 The parameters given in the image satisifes the congestion scenario i.e incoming traffic should be greater than outgoing link's bandwidth. Using C++ code we simulated this topology in ns-3.

4. Learning About Congestion Control: We learnt about different TCP flavours like TCP tahoe, Reno, NewReno, Cubic etc. and how they handle congestion, their disadvantages and disadvantages, use cases etc. We also learnt about how routers handle congestion different packet dropping policies and how they impact current internet.

5. Generating Data: By simulating given network topology we generated that can be used for further analysis.

6. Understaing data: At this phase, we basically analysed the generated data what different inferences we can generate from it.

7. Finding Optimal Parameters: We propose using data driven approach to calculate β, α and k values, and from that we can calculate value of q_{th}.

8. Updating the metric: In this phase we will be implementing our approach in ns-3. Comparing our results with with general congestion scenario and check if our approach fairs better or worse.

5 Theoretical/Numerical/Experimental findings

The result is generated using simulation for the given topology (4.1). Trace file at router 1 was collected as well as change of cwnd size with time for each client node was also collected.

Some of the result is discussed below:

1. This graph (5.1) shows how cwnd size varies with time for a single node. You can see in this graph

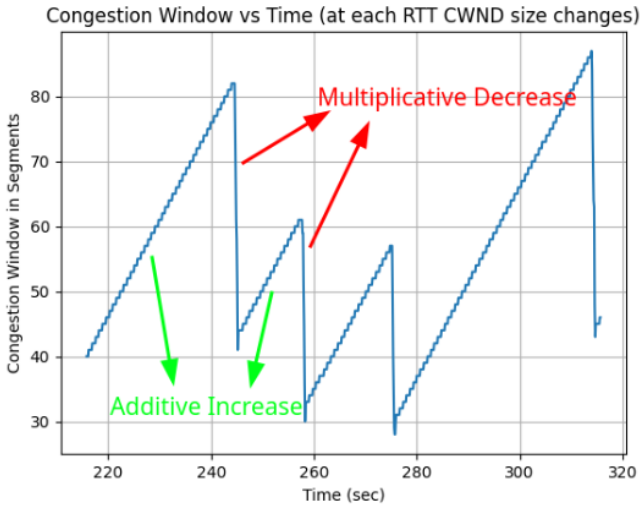

Figure 5.1: Congestion window (w) vs time for one node

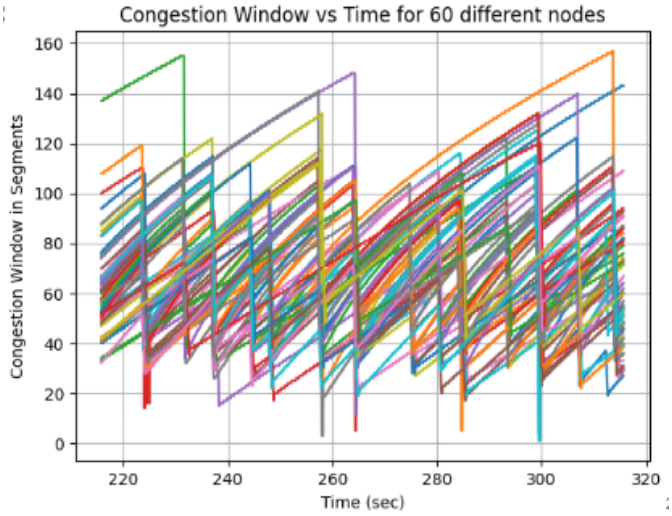

Figure 5.2: Congestion Window (w) vs time for all nodes

how additive increase and multiplicative decrease (AIMD) works in congestion avoidance phase of TCP. Size of cwnd either increases linearly or decreases multiplicatively with each RTT in congestion avoidance phase.

2. This graph (5.2) shows cwnd size vs time for all 60 nodes. This graph shows how global syncronization leads to bandwidth underutilization. Global synchronization means either all nodes send traffic simultaneously or back off simultaneously.

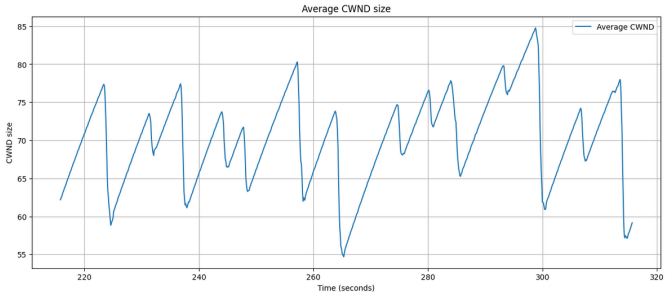

Figure 5.3: Average Congestion Window ($w*$) vs time

3. This graph (5.3) show average cwnd (w^*) vs time. w^* is average of w for all nodes with time.

6 Summary and Future plan of work

In our project we aim to propose a solution to widely encoutered situation in today's internet connected world, Network Congestion. Congestion wastes important resources time, computing power etc. If not handled properly it can lead to collapse of whole internet infrastructure. It occurs in routers, when incoming traffic is greater then bandwidth of outgoing link. Routers have finite buffer that they use to contnuously send data thus utilizing full bandwidth. Most of recent work is focused on making end points more intelligent and handling congestion mainly through them. Active Queue Management (AQM) serves as a way for router to control congestion as well as notify end points that a congestion has occured or it is likely to occur if they don't lower their sending rates. Random Eearly Detection randomly starts to drop packets as queue get full. While Thresold based queueing policy starts to drop packets when queue size crosses a certain threshold. In our project we aim to modify threshold based queueing policy such that it ensures stability as well as greater throughput thus more utilization of bandwidth. We will be using data driven approach to learn queue threshold parameter.

Future work includes refining our approach more as well as fixing inconsistencies in the data generated. Using different approaches to calculate α, β and k parameter. Implementing this new AQM policy in ns-3 and comparing its result with current widely used methods.

References

[1] Maximilian Bachl. *Fair Queuing Aware Congestion Control*. June 2022. DOI: 10.48550/arXiv.2206.10561.

[2] Ethan Blanton, Dr. Vern Paxson, and Mark Allman. *TCP Congestion Control*. RFC 5681. Sept. 2009. DOI: 10.17487/RFC5681. URL: https://www.rfc-editor.org/info/rfc5681.

[3] Wesley Eddy. *Transmission Control Protocol (TCP)*. RFC 9293. Aug. 2022. DOI: 10.17487/RFC9293. URL: https://www.rfc-editor.org/info/rfc9293.

[4] S. Floyd and V. Jacobson. "Random early detection gateways for congestion avoidance". In: *IEEE/ACM Transactions on Networking* 1.4 (1993), pp. 397–413. DOI: 10.1109/90.251892.

[5] Andrei Gurtov et al. *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 6582. Apr. 2012. DOI: 10.17487/RFC6582. URL: https://www.rfc-editor.org/info/rfc6582.

[6] Shao Liu, T. Basar, and R. Srikant. "Exponential-RED: a stabilizing AQM scheme for low- and high-speed TCP protocols". In: *IEEE/ACM Transactions on Networking* 13.5 (2005), pp. 1068–1081. DOI: 10.1109/TNET.2005.857110.

[7] Rong Pan et al. "PIE: A lightweight control scheme to address the bufferbloat problem". In: *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*. 2013, pp. 148–155. DOI: 10.1109/HPSR.2013.6602305.

[8] Lisong Xu et al. *CUBIC for Fast and Long-Distance Networks*. RFC 9438. Aug. 2023. DOI: 10.17487/RFC9438. URL: https://www.rfc-editor.org/info/rfc9438.