Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN

Ingeniería de Telecomunicación

TRABAJO FIN DE MÁSTER

EVALUATION OF ACTIVE QUEUE MANAGEMENT (AQM) MODELS IN LOW LATENCY NETWORKS

Alumno/Alumna	*Sanz Tobias Xabier*
Director/Directora	*Pinto, Charles and Kutka Eduardas*
Departamento	*DEPARTMENT OF NETWORKING*
Curso académico	*2018/2019*

Bilbao, 1, Julio, 2019

VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

UNIVESRITY OF THE BASQUE COUNTRY
FACULTY OF ENGINEERING
DEPARTMENT OF NETWORKING

Final Master Thesis

Evaluation of Active Queue Management (AQM) Models in Low Latency Networks

Done by:

Xabier Sanz signature

Supervisor:
Eduardas Kutka
Charles Pinto

Vilnius
2019

I would like to thank Vilnius University and Basque Country University
for their collaboration with the project.
Thanks to the far ones.
Thanks to the close ones.

"Quiero bailar ahí arriba como en un ultraligero"

Contents

List of Figures 6

List of Tables 7

Abstract 8
- Keywords 8

Resumen 9
- Palabras Clave 9

Introduction 10

1 Background 11

2 Scope and Objectives of work 13

3 Benefits that work brings 14
- 3.1 Economic benefits 14
- 3.2 Social benefits 14
- 3.3 Environmental benefits 15
- 3.4 Technical benefits 15

4 State of Art 16
- 4.1 Classic TCP Issues for Low Latency 16
  - 4.1.1 Slow Start 18
  - 4.1.2 Congestion avoidance 19
  - 4.1.3 Bufferbloat 19
- 4.2 Queue Management 21
  - 4.2.1 Traditional Queue Management 21
  - 4.2.2 Active Queue Management 22

5 Analysis of alternatives 23
- 5.1 RED 23
- 5.2 CoDel 27
- 5.3 PIE 29
- 5.4 FQ_CoDel 32
- 5.5 ABE 34
- 5.6 Decision Criteria 36
  - 5.6.1 Comparison in Online Game Latency 36
  - 5.6.2 Comparison in VoIP Latency 38
  - 5.6.3 Comparison in TCP bulk upload 38
  - 5.6.4 Comparison in general features 40
  - 5.6.5 Final Deicision 41

6 Proposed solution **42**
6.1 CoDel in detail 42
6.1.1 Building Blocks: "Bad" and "Good" queues 43
6.1.2 Queue control components 43
6.1.3 Non-Starvation 45
6.1.4 Overview 46
6.2 FQ_CoDel in detail 47
6.2.1 Flow Queuing 47
6.2.2 Scheduler 48
6.2.3 Differences from CoDel 49
6.3 ABE: Alternative Backoff with ECN 50
6.3.1 Explicit Congestion Notification (ECN) situation 50
6.3.2 AIMD problem 50
6.3.3 Solution 51
6.3.4 Overall 52
6.4 Simulations 53
6.4.1 Topology 53
6.4.2 Scenarios 54
6.4.3 Test's feedback 55
6.5 Results 56
6.5.1 No congestion 56
6.5.2 TCP congestion 62
6.5.3 TCP + UDP congestion 71
6.5.4 Overall Results 80

7 Methodology followed in the development of the work **82**
7.1 Description of tasks, phases, teams or procedures 82
7.1.1 Project Supervision 82
7.1.2 Prior Preparation 82
7.1.3 Preliminary Analysis 82
7.1.4 Study Methods 82
7.1.5 Case Study 82
7.1.6 Thesis composition 82
7.2 Gantt Chart 83

8 Economic Aspects **86**
8.1 Budget 86
8.1.1 Work hours 86
8.1.2 Repayments 87
8.1.3 Total Cost 87

Conclusions and Recommendations **88**

References **89**

References **89**

Appendices **91**

A PRODUCT SPECIFICATION **92**

A.1 Script usage 92

A.2 Configuration of ns3 parameters 93

List of Figures

1 Main TCP‘s Latency Causes: Queuing, Distance and Packet Loss 11
2 Bandwidth and Latency [11]. 16
3 Bandwidth and Latency Comparison in Web Page [2]. 17
4 TCP Congestion Mechanism [11]. 19
5 Drop Tail Mechanism 21
6 Synchronization of Tail-Drop [22]. 22
7 RED Algorithm 24
8 Four FTP Connection with RED 25
9 (1) First Graph Shows RED Algorithm Probability Dropping, (2) Second Graph Shows the Probability Dropping of Drop Tail Mechanism 25
10 CoDel Algorithm Simplified 28
11 PIE Design [1] 29
12 RED vs PIE Varying Link Capacity [1] 30
13 RED vs PIE Varying Traffic [1] 31
14 FQ_CoDel Queues Model 33
15 Gaming Packet Latency [29] 37
16 MOF in VoIP [29] 38
17 TCP Bulk Performance [29] 39
18 CoDel Algorithm Simplified 46
19 FQ_CoDel Queues Model 49
20 TCP Sawtooth Problem. 51
21 AQM plus ECN Proposal. 51
22 CDF of Queueing Delay in TCP Reno and AQM CoDel (10 Mbps). 52
23 Dumbbell Simulation Topology 53
24 Throughput Drop Tail Scenario 1 57
25 RTT Drop Tail Scenario 4 57
26 Throughput Drop Tail Scenario 2 58
27 RTT Drop Tail Scenario 2 59
28 RTT Drop Tail Scenario 3 60
29 RTT Drop Tail Scenario 3 (zoomed) 60
30 Throughput Drop Tail Scenario 3 61
31 Sojourn Time Drop Tail Scenario 4 63
32 Packets in Queue Drop Tail Scenario 4 63
33 Throughput Drop Tail Scenario 4 64
34 RTT Drop Tail Scenario 4 64
35 Sojourn Time CoDel Scenario 4 65
36 Packets in Queue CoDel Scenario 4 65
37 Bytes in Queue CoDel and FQ_CoDel Comparison 66
38 RTT Drop Tail Scenario 5 67
39 RTT CoDel and FQ_CoDel Scenario 5 67
40 Sojourn Time Comparison Drop Tail and (FQ)CoDel at Scenario 5 68
41 Bytes in Queue Comparison in Scenario 5 68
42 RTT of the Scenario 6 69
43 Bytes in Queue Comparison in Scenario 6 70

44 Throughput CoDel Scenario 7 71
45 Throughput FQ_CoDel Scenario 7 72
46 RTT CoDel Scenario 7 72
47 RTT FQ_CoDel Scenario 7 73
48 Bytes in Queue Scenario 7 73
49 RTT Drop Tail Scenario 8 75
50 RTT CoDel Scenario 8 75
51 RTT FQ_CoDel Scenario 8 76
52 Bytes in Queue Comparison Scenario 8 77
53 RTT CoDel Scenario 9 77
54 RTT FQ_CoDel Scenario 9 78
55 Packets in Queue Scenario 9 79
56 Sojourn Time Scenario 9 79
57 Throughput Improvement for RTT=50ms 80
58 Throughput Improvement for RTT=100ms 80
59 Throughput Improvement for RTT=400ms 81

List of Tables

1 Online Game performance comparison 37
2 General Features Table 40
3 Scenario 1 Results Values 56
4 Scenario 2 Results Values 58
5 Scenario 3 Results Values 59
6 Scenario 4 Results Values 62
7 Scenario 5 Results Values 66
8 Scenario 6 Results Values 69
9 Scenario 8 Results Values 74
10 Work Breakdown Structure 84
11 Work Hours Cost 86
12 Cost by Phases 86
13 Work Hours Cost 87
14 Total Cost 87
15 ns3 Parameters 93

Abstract

Low latency networks require the modification of the actual queuing management in order to avoid large queuing delay. Nowadays, TCP's congestion control maximizes the throughput of the link providing benefits to large flow packets. However, nodes' buffers may get fully filled, which would produce large time delays and packet dropping situations, named as bufferbloat problem. For actual time-sensitive applications demand, such as VoIP, online gaming or financial trading, these queueing times cause bad quality of service being directly noticed in user's utilization. This work studies the different alternatives for active queue management (AQM) in the nodes links, optimizing the latency of the small flow packets and, therefore, providing better quality for low latency networks in congestion scenarios. AQM models are simulated in a dumbbell topology with ns3 software, which shows the diverse latency values (measured in RTT) according to network situations and the algorithm that has been installed. In detail, RED, CoDel, PIE, and FQ_CoDel algorithms are studied, plus the modification of the TCP sender's congestion control with Alternative Backoff with ECN (ABE) algorithm. The simulations will display the best queueing times for the implementation that mixes FQ_CoDel with ABE, the one which maximizes the throughput reducing the latency of the packets. Thus, the modification of queueing management with FQ_CoDel and the implementation of ABE in the sender will solve the bufferbloat problem offering the required quality for low latency networks.

Keywords

latency; TCP; AQM; bufferbloat; ABE ; FQ_CoDel

Resumen

Evaluación de los modelos AQM en Redes de Baja Latencia

Las redes de baja latencia requieren la modificación de la actual gestión de las colas con el fin de eludir los extensos tiempos de retardo. Hoy en día, el control de congestión de TCP maximiza el rendimiento (throughput) del enlace otorgando beneficio a los grandes flujos de datos, sin embargo, los buffers son plenamente cargados generando altos tiempos de retardo y fases de retirada de paquetes, llamada a esta situación el problema de Bufferbloat. Par las aplicaciones contemporáneas como las llamadas VoIP, los juegos on-line o los intercambios financieros; estos tiempos de cola generan una mala calidad de servicio detectada directamente por los usuarios finales. Este trabajo estudia las diferentes alternativas de la gestión activa de colas (AQM), optimizando la latencia de los pequeños flujos y, por lo tanto, brindando una mejor calidad para las redes de baja latencia en situaciones de congestión. Los modelos AQM han sido evaluados en una topología 'dumbbell' mediante el simulador ns3, entregando resultados de latencia (medidos en RTT) de acuerdo con la situación del enlace y el algoritmo instalado en la cola. Concretamente, los algoritmos estudiados han sido RED, CoDel, PIE y FQ_CoDel; además de la modificación del control de congestión TCP del emisor denominada ABE (Alternative Backoff with ECN). Las simulaciones que mejor resultados ofrecen son las que implementan combinación de FQ_CoDel con el algoritmo ABE, maximizando el rendimiento y reduciendo la latencia de los paquetes. Por lo tanto, la modificación con FQ_CoDel en las colas y la de ABE en el emisor ofrecen una solución al problema del Bufferbloat altamente solicitada por las redes de baja latencia.

Palabras Clave

latencia; TCP; AQM; bufferbloat; ABE ; FQ_CoDel

Introduction

Bufferbloat is defined as the exceed buffering of packets in nodes queues, generating large times of latency values. The demand of time-sensitive applications, such as VoIP, video conferences or online gaming; have significantly increase, being crucial low latency networks for user's quality of experience. However, Bufferbloat problem in combination with traditional Transport Control Protocol's (TCP) congestion control mechanism, makes this requirement hard to fulfill. Packets need to be delivered quickly, with the minimum amount of delay between the sender and the receiver, taking into account distance, packet loss and queuing delay. In particular, queuing delay is the main point of this thesis, being the spot where the bufferbloat occurs. As a solution, this work evaluates several alternatives of active queue management (AQM) models, those who correct the latency problem when congestion is created at the node.

In queuing, Drop Tail technique is the classic model of queue management, though it does not offer the time requirements that actual applications call for. Drop Tail, under congestion situation, starts throwing all the packets that arrives after the buffer is full. This way, the throughput of the path is maximized, but the delay harms dramatically to small flows that usually are time-sensitive ones. Small flows, called as 'mice', are the ones used by low latency applications, whose fight against traditional large flows, 'elephants' named. AQM models search for the delay priority for 'mice' flows, managing the queue in order to maximize the throughput and minimize the delay across the path.

Evaluation is performed by a simulation of the AQM techniques under diverse scenarios, in search of the algorithm that better adapts itself to the traffic. The evaluation is done with ns3 simulator, the one highlighted by great majority of the researchers. Simulations feedback sojourn times (the time that a packet spends inside the queue), throughput, round trip times (RTT) and other values; whereby the algorithms comparison will be done.

The results of the work, stand out the combination of FQ_CoDel with Alternative Backoff with ECN (ABE) sender's modification. The study clearly promotes the use of AQM in queue management, necessary being for actual and future networks, beside the use of explicit congestion notification (ECN) marking in actual flows paths. In conclusion, the thesis evaluates different AQM models towards looking for the optimization of the time-sensitive applications in low latency networks.

1 Background

Transport protocols are used by the network nodes to carry applications information around the Internet, but also they send additional information to provide features like checking, correction or recovery from errors that happens in the transmission. There is a large number of transport protocols, however, the three representatives are User Datagram Protocol (UDP), Reliable Data Protocol (RDP), and Transmission Control Protocol (TCP). This thesis will be focused in the latency created by the Transmission Control Protocol.

Traditional Transmission Control Protocol (TCP) is used by most of the actual Internet protocols. It grants a reliable, connection-oriented and transport protocol for transaction oriented applications, controlling the data transmission over the network via sender-side congestion-window and receiver side advertised network [8]. Traditional TCP was developed for reliable packet delivery, providing good quality of service (QoS) to bulk applications, such file transfer protocol (FTP). However, today´s Internet applications are focused in timely packet delivery, being the latency one of the strongest point to optimize for applications like interactive web browsing, online games or audio/video conferencing, among others.

Latency is defined as the time that a packet needs to arrive from a sender to a receiver, in other words, is a measure of the responsiveness of an application. Is additive in nature and accumulates over the communication session or application task, therefore, even slight unnecessary delay per transfer adds up to considerable overall delay. Thus, it is necessary to analyze the main causes of TCP´s latency that has a huge impact on time-sensitive applications [16]

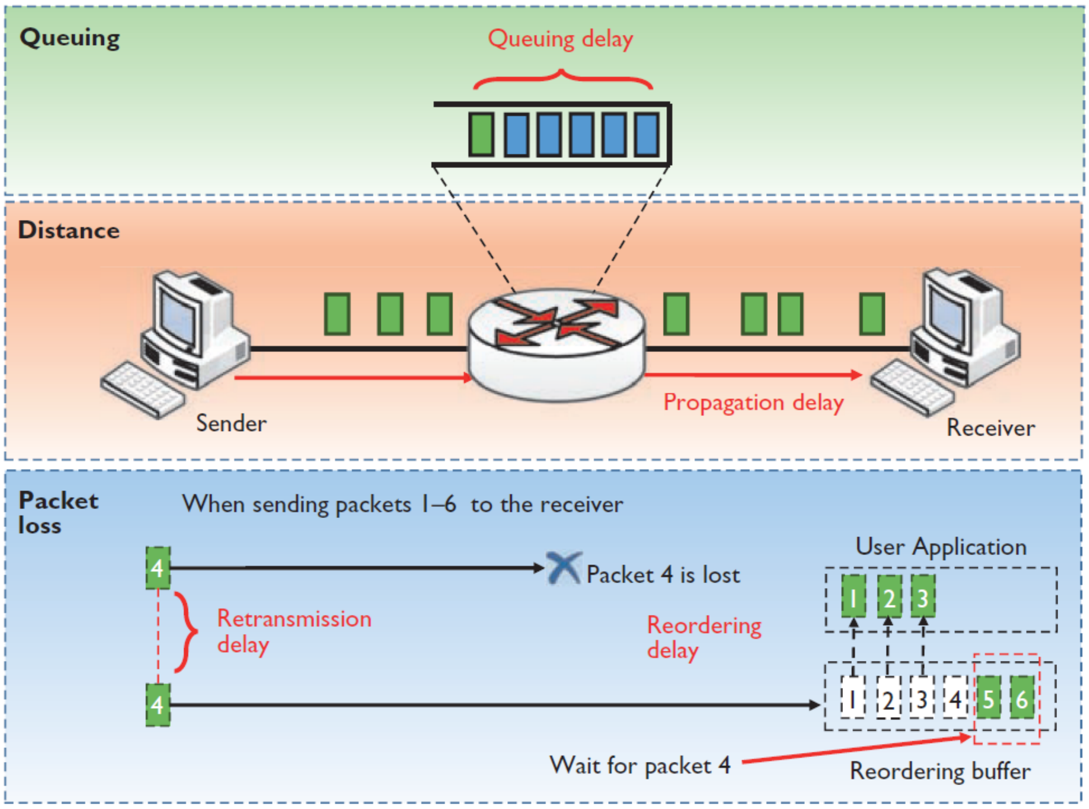


Figure 1. Main TCP‘s Latency Causes: Queuing, Distance and Packet Loss

Distance

The distance between one point to another causes propagation delay, being this the length of time taken for the bits to reach the destination. It depends on the distance and the speed of propagation, thusly, as both parameters are constant in all sessions, propagation delay is predictable and stable. [16] explains that the typical inter-country propagation delay can be around 100 milliseconds.

Queuing

Nodes between the sender and the receiver, such router, and switches, use queues for the proper handle of the packets that arrive. These queues are used when is necessary to absorb bursty traffic that often occurs, besides to reshape the traffic patterns. Nevertheless, the queue generates a delay, which could be of hundreds of milliseconds. The thesis will focus the research in the treatment of the network devices queues, analyzing the main problems and solutions.

Packet Loss

Finally, fast-retransmission is a technic used by TCP that provides a solution to packets that have been lost during the process of the transmission. Inside the fast-retransmission, exists two types of addition delay: the retransmissions delay and the reordering delay.
On the one hand, retransmissions delay is triggered by a timeout or by an activator. The time to recover the lost packet causes a delay, in default conditions, around 200 milliseconds.
On the other, reordering delay occurs when TCP retransmits a packet and the delivery order breaks the correct order of packets. TCP provides reliable, ordered, and error-checked delivery, being necessary the in-order delivery of these subsequent packets. For that, the packets will stay in the system buffer until the delayed packet has arrived.

The three explained causes, shown in Figure 1, are created only from one TCP connection, withal, each application could generate around half dozen connections, doing for each one 3-way handshake, DNS lookup, and connection establishment processes.
According to [3] , downloading one short web page, 6 connections are created with 50 ms of latency for each one. If more than one web page, or other applications are used, it is easy to generate 40 TCP flows, what that entails $40 * 50ms = 2seconds$ of delay. For a human, each click in a link of the web page will translate in 2 seconds of waiting time, doing the use of the navigation intermittent and unnatural.

Against this background, it is crucial to fix the traditional transport protocols in order to provide an appropriate performance on nowadays applications. Latency-sensitive applications (such interactive web, voice, online gaming, instant messaging or remote desktops) have grown in demand of the users, requiring fast transmissions and the decreasing of the delay to the minimum possible. Further, five generation (5G) network will demand, in a nearly future, ultra low latency (ULL) networks, making the implementation of low latency networks more necessary.

2 Scope and Objectives of work

The main focus of this thesis is to evaluate solutions for the actual latency values in Internet's TCP transport protocol. The goal is to select the optimal solution that will provide near zero-latency times without forgetting the coexistence with the traditional protocols. The thesis will essentially target in queue management solutions, analyzing the behavior of queues and the filling process.

The methods to analyze will share a point in common, Active Queue Management (AQM) technique which manages queues to achieve certain queue loss and latency characteristics by proactively marking or dropping packets. This technique is the proposed one by the Reducing Internet Transport Latency (RITE) [21] group, who has been researching and publishing about the transport latency in the last years (in collaboration with IEEE and IETF).

The evaluation of the solutions will be developed in scenarios where data-flows would be mixed by large (elephants) and light (mice) flows, in order to recreate actual networks as real as possible. The characteristics of both traffic types will be taken into account, for different paths and congestion types.

After finding and selecting a solution, it will be detailed analyzed, centering in the benefits that it offers to the problem. In addition, areas for possible improvement would be evaluated for further considerations. Finally, the future of latency networks will be discussed with the possible evolution of Internet transport protocols.

In summary, the present work explores the different queue management alternatives that provide benefits to the actual latency, selecting one that minimizes the latency for low latency networks. Coexistence with the actual protocols is a must in the research, the future will be composed of heterogeneous networks, so it is indispensable to offer that skill. The selected proposal will be detailed analyzed and evaluated for a possible improvement. The thesis will continuously keep in mind that actual network transport protocols are not prepared for the near future remaining the needed evolution for the demand of new services and usages.

3 Benefits that work brings

3.1 Economic benefits

Adapting nodes queues to improve the main Internet´s transport protocol will offer divergent companies developing new services and equipment bringing notorious economical benefits. First, equipment manufacturers would be able to increase their node's speed, adapting them to the new generation network, with low latency premises and improving general capabilities without investing too much in new hardware. Second, service providers, particularly time-sensitive services like interactive web, voice, online gaming or financial transactions, would be able to attempt better quality of their services increasing the demand of those ones.

In particular, speaking about economic benefits, low latency financial networks would be the most benefited area, where the minimum delay downgrade will translate in billions of profits [27]. The financial environment is ruled by the "fast" information gathering and the actions prompted by this information. Hence, with lower latency, bigger the profits of the financial company. Being faster than other traders can create profit opportunities by enabling a quick response to news or market activity. Today's markets are based in "millisecond environment", where is placed nowadays latency [17].

In consequence, bringing the actual latency to a new generation low latency network would carry substantial economic benefits to all Internet companies.

3.2 Social benefits

Social benefits are usually related to economic benefits. By the emerge of new services related to the increase in companies' profits, society will demand more services, generating profits to the companies, who will develop new technologies and services.

In addition, the industrial world requires tens of milliseconds for fast control, however, the delay of traditional transport protocols is bigger than that rate. Improving it, connections between emergency cars, machines, and clouds would be accelerated, that is to say, decreasing the latency between emergency points automatically will provide an important social advance. Usually, these are small data transmissions, with quick response and big priority inside the network. Vital information could be stored in packets, becoming one of the most important benefits by reducing the delay-time near to zero as possible.

Social benefits are tightly related to the economic benefits and low latency networks will benefit these two points notably.

3.3 Environmental benefits

Low latency networks do not involve directly an environmental benefit, however, the services that will generate this network will help in several environmental warning alerts. Increasing the velocity of data transmission between two nodes combined with the Internet of Things (IoT) equipment would permit to detect faster natural disasters such as floods or forest fires.

Therefore, speaking about environmental benefits, the combination of low latency networks with IoT technologies, will develop a large amount of services and systems that would help in the detection and prevention of climate changes.

3.4 Technical benefits

Changing the point of view of the traditional transport protocols, giving weight to de quick delivery instead of the reliable deliver of packets, will adapt actual networks to the future services, making a huge Internet's structure evolution. Because of that, these work's technical benefits are the most relevant.

All the technical benefits will be shown along the work, however, this section will present an overview of the most important ones.

First, solutions will discuss the size of buffers in the access network nodes, large buffers are more reliable but increase packet delivery latency. Contrarily, small buffers drop earlier data packets, causing retransmissions. Finding the perfect size and combining it with other techniques will be one of the technical benefits for managing nodes queues.

Second, AQM will help in latency values for traditional TCP´s congestion protocol. AQM will notify sender before the buffer is full, improving latency and jitter and reducing the level of burstiness in lossless which turns leads to the de-synchronization of the flows.

There are many technical benefits, however, now have been shown which ones are going to solve the main problems, generating, precisely, the main benefits of low latency networks. The evolution of the network will bring changes in all the areas, creating a faster Internet and bringing the next step to the 5G networks.

4 State of Art

This section will analyze the state of art of today´s TCP stage related to low latency networks. First, it explains a general point of view of classic TCP latency drawbacks, defining the issues and main problems of traditional TCP techniques against the delay sensitive applications. In addition, bufferbloat problem will be described, relating it with the Queue Management techniques definition. Finally, solutions and alternatives of the actual state will be revealed, bringing the first strokes of the thesis proposed solution.

4.1 Classic TCP Issues for Low Latency

TCP is the major protocol for a gross portion of Internet traffic. It controls the congestion of the network regulating the quantity of data injected in the network per time. To do this, it has two operating modes: slow start and congestion avoidance [15]. But first, it is needed to explain the bandwidth-latency problem.

Bandwidth is defined as the amount of data that can be transferred via a logical or physical path at a given time. In large transmissions (elephants) latency and bandwidth are correlated, in other words, greater bandwidth means a greater pipe to sent data in a parallel way. However, web browsing, VoIP, online gaming, ... applications do not send much data, being limited by latency, not bandwidth. Thus, upgrading connection bandwidth does not help.

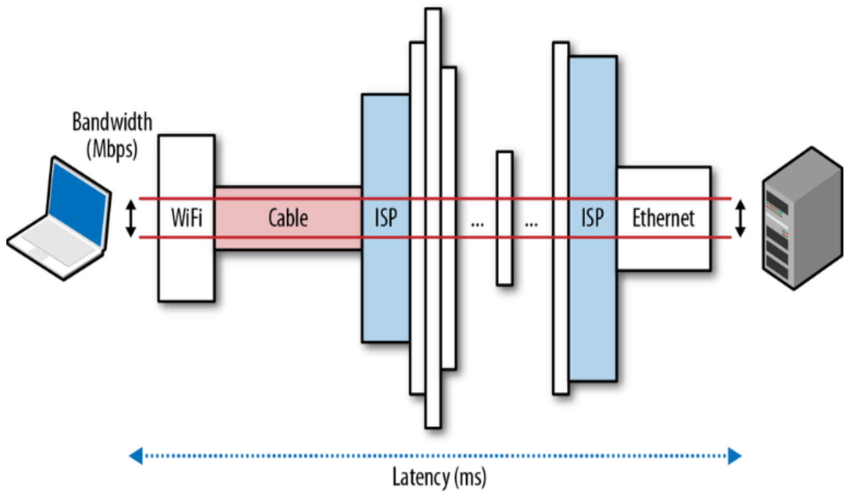


Figure 2. Bandwidth and Latency [11].

Down below, is explained an example of this problem according to Belshe [2] study in order better understand of the bandwidth-latency issue in time-sensitive applications. Web pages are composed by small applications-limited flows, that generate hundreds of flows asking small resources to different places of the net. The round trip time (RTT) represent the latency in study, being the time that needs a packet to go and come back from the sender to the receiver.

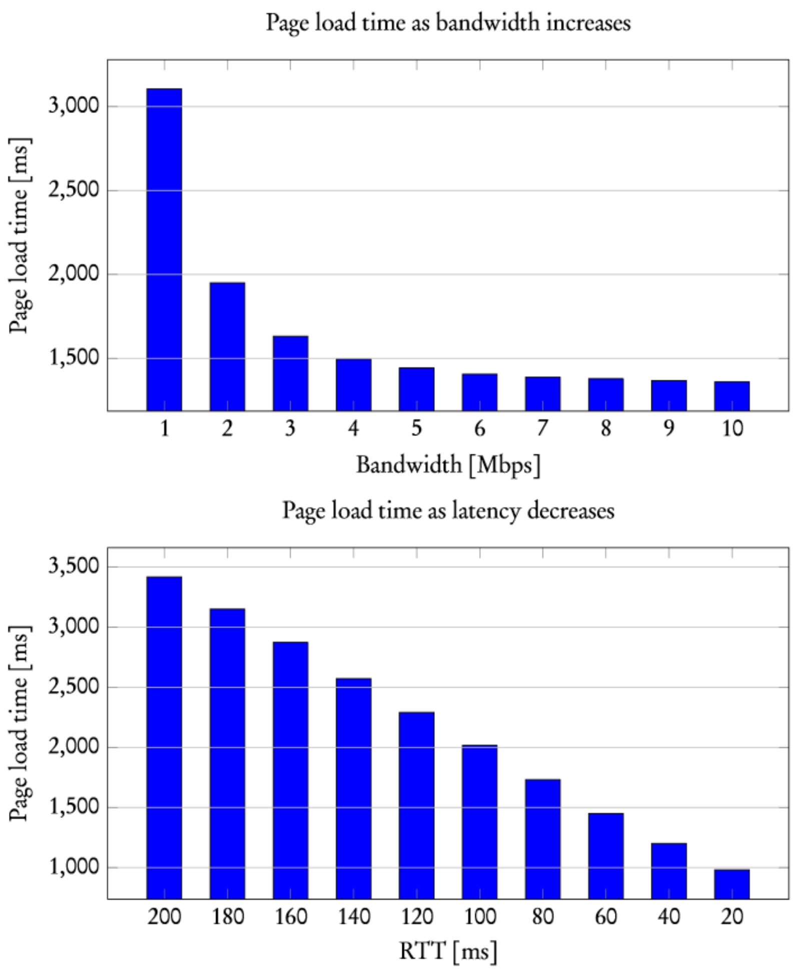


Figure 3. Bandwidth and Latency Comparison in Web Page [2].

Figure 3 shows two plots, which compares the load time of a page depending on the variability of the bandwidth or latency. The main point of this comparison is to identify that the increase of bandwidth is evident during the first bandwidth rises (from 1 Mbps to 2 Mbps), but latency increment follows a linear decrement of page load time. Therefore, for small packet flows (mice), the increase of bandwidth does not help as much as the increase of performance of latency.

4.1.1 Slow Start

Despite being thought that the increase of bandwidth will improve network performance, as it is shown in the previous example, it only brings a faster transmission, not necessarily being a solution for latency necessities. TCP start-up mechanism, by contrast, can contribute to latency overall value. At the beginning of a new connection, both, sender and receiver, do not know the available bandwidth, so they need a mechanism that estimates and adapt the value of their transmission speed.

Slow-start is an algorithm used by TCP protocol that provides a mechanism to probe network available capacity, in order to become congested it with an inappropriately large burst of data [13]. In a new connection, the three-way handshake is performed and both sides advertise their receive window (*rwnd*) sizes. However, they need to estimate the capacity of the connection, being this the propose of slow-start. The server initializes a congestion windows (*cwnd*) variable per TCP flow, but it is not exchanged to the client. Thus, the client starts with a system preset congestion window value (1, 4 or 10 segments). The minimum between *cwnd* and *rwnd* will be the data in flight (not ACKed data) and congestion window will be increased per every ACK received (for every ACKed packet, two new packets can be sent). This is the exponential growth that permits both sides to converge to the available bandwidth.

The time needed by a connection to reach a specific throughput (N) (bandwidth level, for example) is established by the Equation 4.1, completely related with the RTT (latency) and initial *cwnd* value.

$$Time = RTT * [log_2(\frac{N}{initcwnd})] \tag{4.1}$$

According to [23] the majority of the flows on the Internet are small, not being large enough to reach the maximum window size. In addition, small packets, because of the slow-start, usually arrives at an empty bottleneck, not being an optimal usage of the network resources. As a result, the slow-start algorithm limits the available capacity of the connection, harming the performance of time-sensitive applications where latency and congestion window size are limiting factors.

4.1.2 Congestion avoidance

TCP's congestion avoidance is used when a packet loss occurs. The explained slow-start mechanism will increase the amount of data sent until it exceeds receivers windows size or a packet lost happens, thus congestion avoidance does not prevent the congestion, it will only detect and correct it.

Congestion means that some node between sender and receiver has a link that could not process in a proper speed the packets and it drops them. TCP, by default, detects these dropped packets and adjust the congestion window in order to decrease the congestion. After that, the sender will start increasing the window until another loss event occurs, generating the famous "sawtooth" TCP pattern.

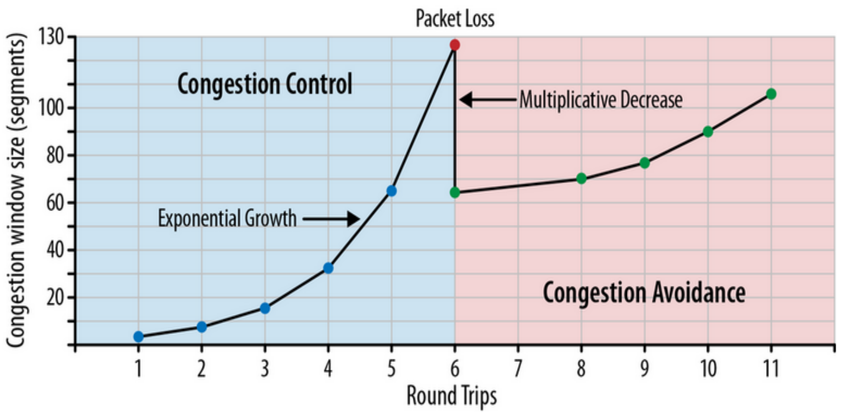


Figure 4. TCP Congestion Mechanism [11].

In summary, congestion avoidance only carries effect WHEN a point between origin and destiny has been ALREADY congested, generating packet loss and, consequently, packet retransmissions. These two situations, have a direct effect in nodes queues, becoming the main point of latency addition. The following sections will study these situations an provide solutions to them.

4.1.3 Bufferbloat

The goal of buffers at network devices is to provide a certain space for packets that egress link could not process as quick as ingress rate. They avoid the discard of packets during the time when the egress link is busy, generally, due to short-term packet burst. In addition, in TCP connections the buffer assures the full usage of the egress link when the sending rate of the sender starts decreasing.

However, the use of large buffers in order to avoid these situations is not the solution for today's Internet. During the last decades, Internet Service Providers (ISPs) have used bandwidth sale argument as the main point of the traffic performance, forgetting the latency guarantees in their Service Level Agreements (SLA). Apart from that, Random Access Memories (RAM) prices have decreased, becoming easy and cheap to install more buffer to the equipment, offering large buffers spaces.

On the other hand, the common rule-of-thumb is used to figure out the optimal amount of buffer needed for a single TCP flow (using NewReno congestion control). It is defined as the product of the capacity (C) of the bottleneck between sender and receiver and the delay (RTT).

$$Buffer = C * RTT \tag{4.2}$$

Rule-of-thumb, also known as the bandwidth-delay product (BDP), assures that a single standard TCP NewReno flow can fully utilize the link, and as is explained in [18] the consequence of inducing a maximum queuing delay to one RTT, increase the total delay to $2 * RTT$ (one of the original RTT and the other due to the buffer delay), affecting directly latency-sensitive applications.

In view of the above, the use of large buffers affects directly to the Internet´s latency referred to as "bufferbloat" effect. The main damage applications are delay-sensitive ones, such as online-games, financial trades or streaming audio-video, where TCP congestion control tend to fill up the buffer, delaying the process of these applications. Plus, the effect of bursts traffic in large buffers cause large latency and jitter to the other flows who share the bottleneck.

4.2 Queue Management

Solving the actual queue management, avoiding bufferbloat, is one of the major concern of the networking researches. Changing the traditional queue management is crucial, offering the best quality in packet treatment and adapting it to the necessities of each one. Next sections will explain the traditional queue management problem and the solution with active queue management techniques.

4.2.1 Traditional Queue Management

Traditional queue management and TCP's congestion avoidance methods are a dangerous combination for latency. As it has been shown, TCP's congestion avoidance used to fill the queue of the buffer in order to achieve the maximum usage of the link, however, with the traditional queue management packets are dropped, generating packet retransmissions and huge latency addition in the flows.

Drop Tail

Drop Tail is the principal in traditional queue management. The mechanism is simple: the buffer accepts packets until it is filled, after that, last packets arrived are dropped. The main problem of this mechanism is that TCP would reduce the sending rate only after the link has become fully congested, notified by dup-ACKs (packet loss). With burst packet arrivals, drop tail will trigger multiple losses, having constantly the buffers full. As it is explained in [30] , multiple papers have offer solutions for manage queues achieving low latency times, being not required traffic class differentiation within the network (Diffserv).

Figure 5 shows the treatment of the packet's drop in Drop Tail algorithm. As mentioned, when the queue is totally filled, drop tail starts dropping all the packets, causing jitter and latency problems.

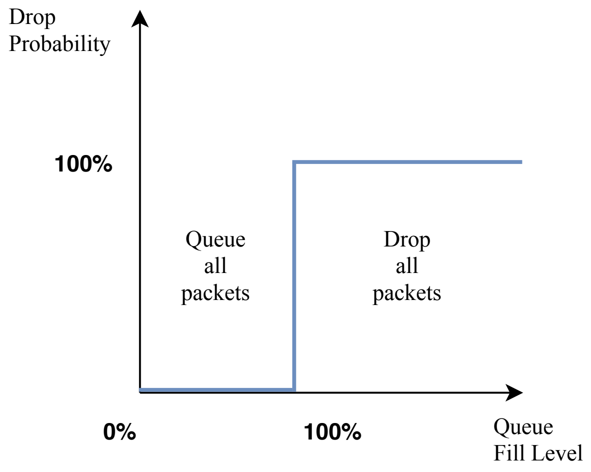


Figure 5. Drop Tail Mechanism

Global Synchronization

Although the focus has tended to be on the management of one flow queue, exist a problem that occurs when N flows transmit together against one network equipment's buffer. Let's imagine that N flows transmit through a router with buffer size B. Each flow, will work with his own congestion window (*cwnd*), affecting individually the congestion notification (dupACKs). The resulting BDP of the affected flow will be smaller as N grows larger [22]. If all the flows work with the same *cwnd*, all flows simultaneously probe the link for extra bandwidth by gradually incrementing their *cwnd*. At one point, the link will enter in congestion growing, at least 1 packet per RTT (in TCP Reno 1packet/RTT, in TCP CUBIC the growth rate is less than 1 RTT).

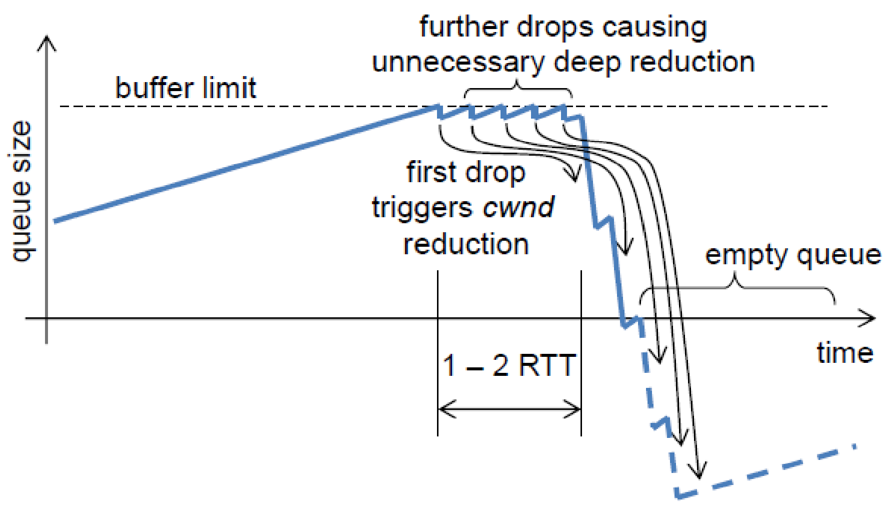


Figure 6. Synchronization of Tail-Drop [22].

Figure 6 shows the process of global Synchronization inside the bottleneck buffer. When the buffer is filled, drop tail queue start dropping all the packets that arrive at the buffer, taking around 1 RTT before the *cwnd* reduction is started. As the period of notifying the *cwnd* reduction is too small, all the flows are notified almost at the same time. Thus, they detect the congestion and start synchronously with the slow-start procedure. At that time, link operates at sub-capacity levels until the buffer is filled again generating again the congestion.

4.2.2 Active Queue Management

Traditional queue management does not provide the requirements to solve bufferbloat problem, maintaining persistently full buffers. Active Queue Management (AQM) offers the solution to the explained problems, notifying the sender to backoff before the buffer is full. The notification could be done via packet-dropping or with the Explicit Congestion Notification (ECN) mark, signaling the TCP sender onset of congestion in an early enough time. Is clear that AQM greatly enhances the jitter and latency values, reducing the burstiness in losses and breaking with the global synchronization issue.

The thesis will research different AQMs models, providing the capabilities of each one and evaluating them against actual traffic flows.

5 Analysis of alternatives

As introduced, Active Queue Management is one of the solutions to improve latency values inside the network equipments' queues. To achieve it, this section will evaluate multiple AQM models, analyzing the proposed technique and estimating the advantages and disadvantages for each one. Finally, the section will present the decision-making criterion and pick the one who is going to represent the thesis proposed solution.

5.1 RED

Traditional TCP protocol does not have explicit getaway feedback for notifying the sender about congestion. Therefore, the trigger that advises congestion is the three duplicated ACKs, created ones the congestion exists in the path, not earlier. Random Early Detection (RED) algorithm [10] was proposed by Van Jacobson 25 years ago, being the first AQM mechanism that advises the sender before congestion is created. In spite of being published a long time ago and able in almost all (nowadays) Linux and Windows version servers, RED has some limitations that difficult the implementation on the Internet.

Is fundamental to select the location where the congestion is detected. As it is explained in [10], that position is the gateway itself. The gateway has a general view of all flows that cross through the buffer, being capable of distinguishing the duration and magnitude of the congestion. Only the gateway has a unified view of the queueing behavior over time; however, the perspective of individual connections could be determined by the packet arrival pattern. In RED, the gateway will sense the congestion and mark or drop the packet to notify it. In addition, due to the single-random marking or dropping, the global synchronization will be avoided.

As will be later explained, RED is based on an average queue (avg) size that monitors the congestion in the queue. Longer-lived congestion will increase the average queue size, resulting in a randomized feedback that will ask to the congestion window to decrease the size. The marking-dropping notification will be done using a randomized algorithm, being the main motivation the avoidance of the global synchronization that results from the simultaneous reduction of many TCP connections windows. When the maximum queue threshold is exceeded (maximum probability), RED will drop every packet, moderating the gateway average size without any cooperating transport protocol from the sender. In addition, RED does not require the same congestion control on all Internet gateways.

Goals

The goals of RED are the following ones:

- Provide congestion avoidance by controlling the average queue size.
- Avoidance of the global synchronization.
- Maintain the average without the cooperation from transport-layer protocols.
- Detect incipient congestion that has persisted for a "long time".

The algorithm

RED algorithm calculates the average queue size and compares it against a minimum and a maximum threshold. If the average size is less than the minimum, no packets are marked. When the average size is between the minimum and the maximum, each arriving packet is marked with a p_a probability, where p_a is a function from the average queue size avg [10]. Finally, when avg is larger than the maximum threshold, every packet are dropped or marked.

```
for each packet arrival
    calculate the average queue size avg
    if minth 5 avg < maxth
        calculate probability pa
        with probability pa :
            mark the arriving packet
    else if maxth 5 avg
        mark the arriving packet
```

The idle period, when there are no packets in the queue, is used to know how much mice packets m could be transmitted. After it, RED use that variable to compare against the small packets that have arrived and know the status of the congestion. Thus, RED is capable to identify burst packets and know every moment the congestion situation and the long flows that are sharing the gateway.

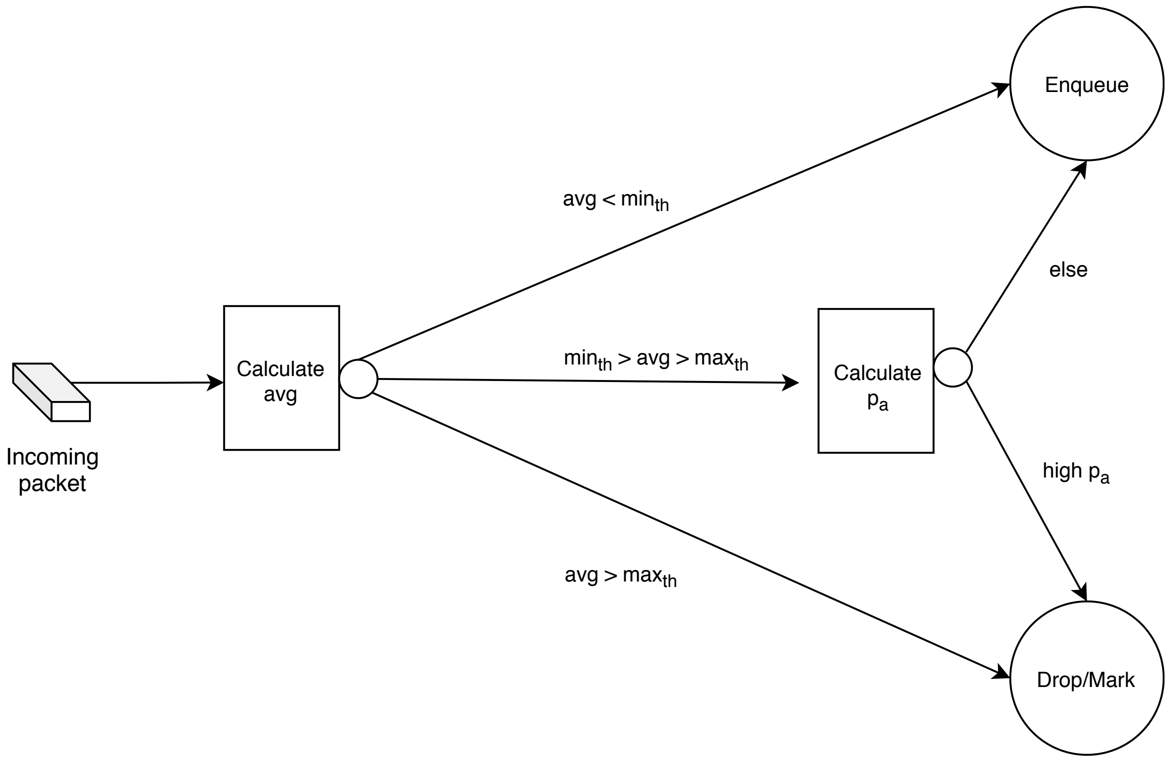


Figure 7. RED Algorithm

The original paper of RED [10], shows several simulations and benchmarks against Drop Tail mechanism. For example, Figure 8 shows a simulation of RED with four FTP connections. Dotted lines are the min and max thresholds, showing the avg size changing slowly across the time until it is stabilized. RED gateways are well suited to provide high throughput and low average delay in high-speed networks with TCP connections that have large windows. However, some weakness will be explained below.

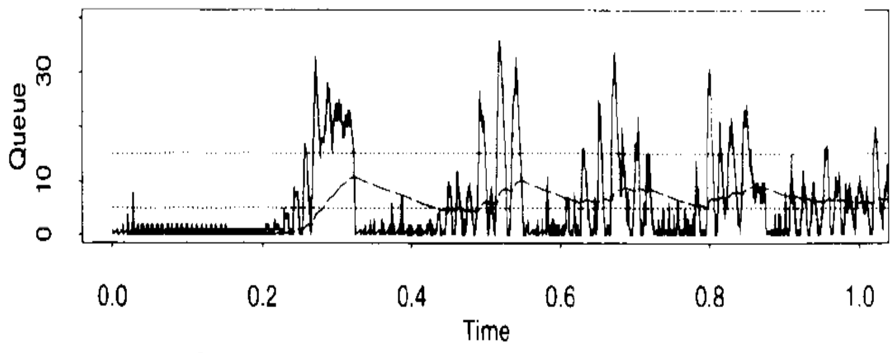


Figure 8. Four FTP Connection with RED

Figure 9 shows the dropping probability of RED against Drop Tail. The threshold of RED makes the AQM model much smooth for dropping, making it before the maximum buffer size is full. As have been explained, it will provide an earlier adaptation to the congestion, taking a proactive role.

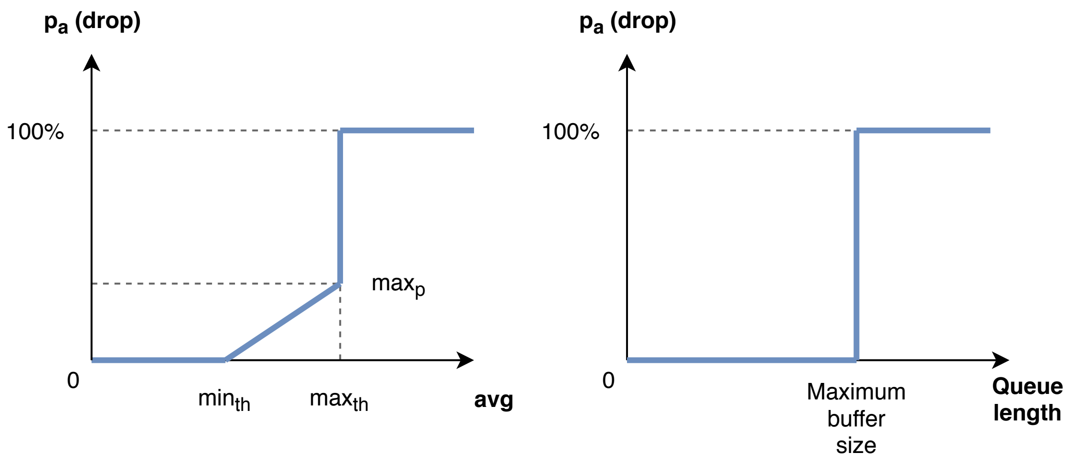


Figure 9. (1) First Graph Shows RED Algorithm Probability Dropping, (2) Second Graph Shows the Probability Dropping of Drop Tail Mechanism

Upshot

RED was the first AQM algorithm developed and proposed, having some issues that following versions and algorithms have corrected. Researchers have analyzed closely the algorithm, evaluating and simulating it in several traffic situations. Next paragraphs explain the issues that the researches have found, based on [12].

The main issue of RED is the parameters "tunning". In spite of avoiding global synchronization, reducing the packet loss and achieving high throughput; RED becomes unreliable in some congestion situations. RED has bad adaptability against network changes, the effortfull tunning of parameters generate large times of dequeue when new flows enter in the queue, being necessary a professional network engineer to tune optimally the algorithm. In addition, many researchers have questioned the QoS and the security provided by using RED in the gateways.

Apart from that, when avg becomes larger than the maximum threshold, RED starts dropping all the packets, forcing senders to decrease sending-rate to the half and generating a notorious throughput misuse at the bottleneck's link. This occurs due to RED's probability function and variables, which ones are not capable to provide the optimal results to the traffic changes. In order to solve the problem, RED algorithms modifications, such Adaptive RED (ARED) [24], have been created, where the avg parameter is maintained between the thresholds almost all the time.

In essence, RED have been the AQM premise model, solving problems like global synchronization, reducing latency in stable traffic situations and working without the help of the transport protocols. However, the evolution of Internet and traffic usage generates poor responses of RED to nowadays latency demands. RED's bad adaptability to networks scenarios and algorithms probability adjustments becomes useless, requiring to modernize the algorithm.

5.2 CoDel

Solving persistently full buffer problem, bufferbloat, is the main concern of AQM methods. RED has several issues with parameter "tunning", being needed other algorithms against the problem. Controlled Delay (CoDel) [14] permits the control of the queue in a parameterless way, providing really good results in latency treatment.

Goals

According to [14], the goals of CoDel are:

- Make AQM parameters no adjustable by operators, users or implementers.
- Be able to distinguish "good" and "bad" queues.
- Control the delay without harming other traffic.
- Adapt to changing link rates.
- Simple and efficient implementation.

The Algorithm

CoDel uses sojourn as monitor value, it is defined as the time that spends a packet inside the buffer, in other words, the difference between the dequeue and queue time of a packet. Using sojourn as monitor value generates a better performance than using queue length. Also, sojourn is used to difference "good" and "bad" queues, applying the dropping/marking treatment only in "bad" queues.

In the end of control the queue, CoDel uses two variables: estimator and target. Estimator uses sojourn time to set the limit between "good" and "bad" queue, being usually the maximum RTT of all connections. "Bad" queues will be those in which CoDel starts with the dropping or marking intervention, providing advantages on delay times. On the other hand, target is the setpoint that tells CoDel when to act. If the algorithms starts dropping to early, the goodput decreases not being an optimal intervention. The goal is a target that maximizes the use while minimizing delay. In future sections these two values will be detailed.

In brief, CoDel algorithm uses time in queue as the monitor parameter. When the queue minimum sojourn is above the target inside a "bad" queue, dropping/marking actions will start. The control loop of the algorithm will try to drive the queuing system from any persistent queue above the target state to a state where the persistent queue is below the target. Dropping period will be variable according to how much packets have been previously dropped, in this way, the goodput is maximized.

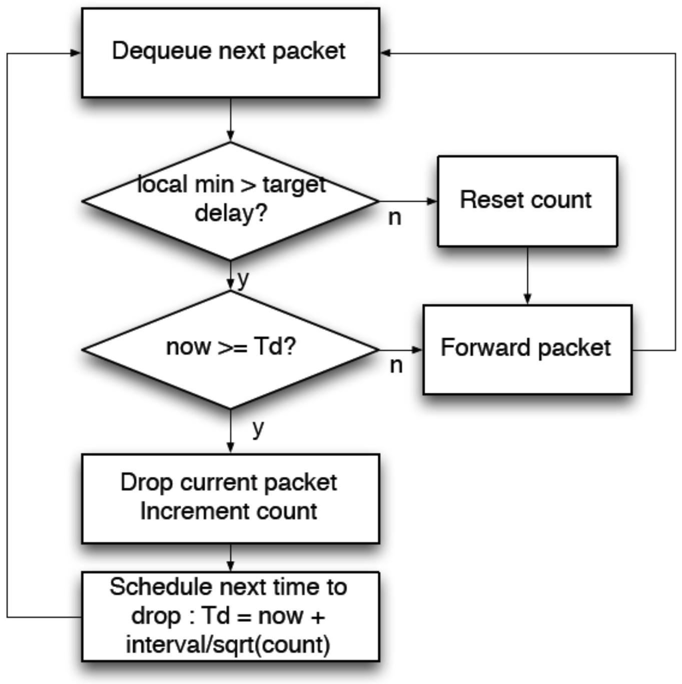

Dequeue next packet
local min > target delay?
n
Reset count
y
now >= Td?
n
Forward packet
y
Drop current packet Increment count
Schedule next time to drop : Td = now + interval/sqrt(count)

Figure 10. CoDel Algorithm Simplified

Upshot

CoDel is one of the AQM methods that best results offers. Many of researchers defend the use of it above other solutions, however, there are some concepts that are not optimized for now. The main point to establish is the deployment of CoDel in every home/small-office access, where the bottlenecks generally are created.

It should be optimized in particular bottlenecks such Wi-Fi, 3G cellular or cable modem, even so, this thesis will target the use of AQM in wired networks, trying to decrease the queue latency to the minimum. Apart from that, CoDel, by itself, does not provides the best results, for this reason, future section will analyze FQ_CoDel, the CoDel application for multiple queues.

5.3 PIE

For addressing bufferbloat problem, several AQM models have been released. Tunning RED is one of the main problems for implementing it, on the other hand, CoDel, consume excessive processing and infrastructure resources, making expensive and difficult to operate. Proportional Integral controller Enhanced (PIE) tries to combine the benefits of both: easy to implement, like RED, and latency treatment, like CoDel.

Apart from that, PIE uses random dropping/marking packets, similar to RED, and congestion detection based on the queueing latency like CoDel (instead of queue length). In addition, the latency government is based on trends, increasing or decreasing the control severity based on the congestion level.

Goals

According to [1] design goals of PIE are:

- Direct control of latency instead of queue length.
- Achieve high link utilization, low latency without losing network efficiency.
- Simple and scalable.
- Stable for diferent network topologies and situations. Design parameters shall be set automatically.

Thus, PIE is the combination of RED and CoDel, mixing the benefits and goals of both and trying to provide the best solution to delay treatment.

The algorithm

Next, PIE working scheme will be explained. It rests in three components: random dropping, periodic drop probability, and dequeue rate estimation.

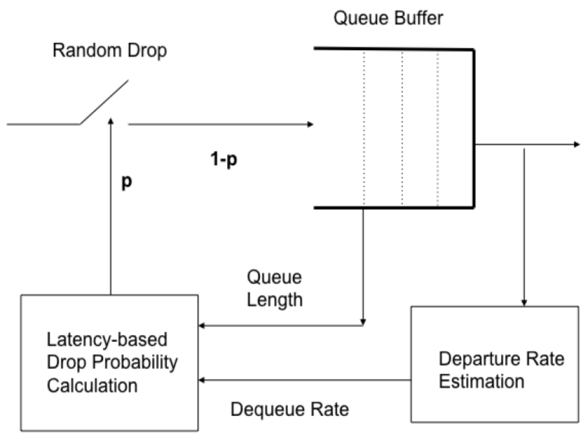


Figure 11. PIE Design [1]

Dropping and marking probability is applied when a packet enters to the device. The probability p will be calculated from an algorithm and applied to the incoming packet. That probability is refreshed periodically according to the following procedure: first, the current queue delay is calculated, secondly, p probability is calculated, and finally, the previous delay is updated.

Drop/Mark probability is calculated according to the queue movement orientation, considering the trend of the queue and delay. For it, α and β parameters are used, α determine the deviation of the current latency and β is an additional adjustment depending if the latency is growing or decreasing.

Departure rate estimation is another point to analyze in PIE. It will be related with the link-sharing and link-capacity fluctuation. Draining rate is calculated when the queue length is larger than a threshold, obtaining a measurement sample. It will be used to update the departure rate and reset counters, making possible to refresh p probability of the first point. As it is shown in Figure 11 "Latency-based Drop probability calculation" is made with the value of queue length and dequeue rate. Plus, because of the periodical p updating burst are tolerated. Any burst that happened in this period of time will be treated without any packet drop, not modifying the drop probability of that time. Apart from that, it is possible to give a precise control of burst to the administrators. For it, PIE offers max_burst parameter similar to the burst tolerance, which, by default is 100 ms, but it could be modified adapting it to the scenario.

Finally, some simulations of PIE against RED will be displayed. The simulations are taken from [1], which describes and evaluates PIE against other AQM methods.

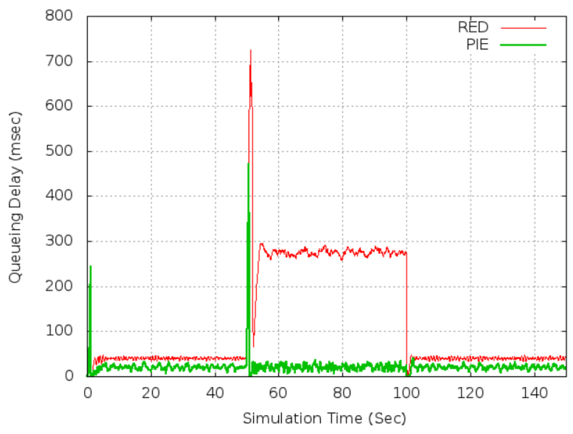


Figure 12. RED vs PIE Varying Link Capacity [1]

Figure 12 shows the reaction of RED and PIE against link capacity decrease. From 0 to 50 s, link capacity is 100 Mbps, then, from 50 to 100 s, the link decreases to 20 Mbps, and finally, from 100 to 150 it grows again to 100 Mbps. RED controls queue length, suffering more the queueing latency when the scenario changes. However, PIE is based in the queueing latency, changing quickly to the new conditions and adapting itself to 20 ms level of latency.

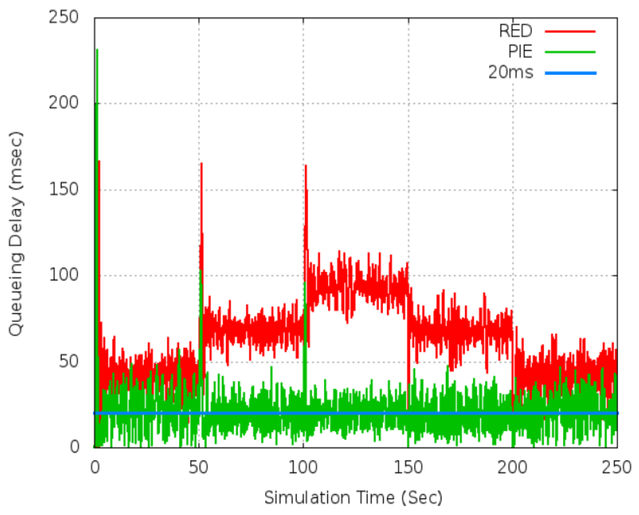


Figure 13. RED vs PIE Varying Traffic [1]

Figure 13 shows the performance of RED against PIE in traffic variation. From 0 to 50 s, traffic load is 10 TCP flows, then, from 50 to 100 s, traffic load is increased to 30 TCP flows, after that, from 100 to 150 s, increased again to 50 TCP flows and from 150 to 250 s reduced to 30 and 10 TCP flows. As we know, RED works with tunned parameters, making a larger queue delay when the load is increased. On the other hand, PIE controls latency optimally adapting automatically the parameters.

Upshot

Proportional Integral controller Enhanced provides control over bufferbloat problem, using random droppings according to the queue trend. It uses auto-tunning for the parameters being effective in diverse scenarios. The main point of PIE is to be light-weight, solving some CoDel problems.

But, according to [19], PIE generates large spikes and queueing delays in large RTTs scenarios. This issue is due to the use of a proportional integral controller that maintains the average queue at the desired level. Thus, dropping probability increases and the utilization of the link decreases. In addition, CoDel has bigger acceptance in networking researchers world, which have being more evaluated and focused for being the next AQM base.

5.4 FQ_CoDel

An overview of CoDel algorithm has been explained previously. It offers a successful control at wide range of bandwidths and flows. In addition, it shorts latency spikes and is able in Linux 3.5 version servers. However, it has issues at very high number of flows. Creating more than one queue for congestion treatment, provides the opportunity to optimize "sparse" ("ants") and TCP mice flows. This solution is provided by FQ_Codel, that could generate up to 64k queues applying independently CoDel mechanism in each queue.

"Ants" concept has been introduced. "ants" are short, usually single, control packets, such SYN/ACK, DNS, DCHP or NTP. Apart from them, some streaming, gaming or VoIP packets have "ants" characteristics too. Thus, Internet is composed by elephants, mice and "ants" flows, being mice and elephants the main load of the traffic. Nevertheless, "ants" are important packets that should be delivered as quick as possible, with a certain priority.

Therefore, FQ_CoDel is a combination of CoDel and multiple queue scheduling scheme that features prioritization and flow isolation. It creates one sub-queue per flow, and each sub-queue uses CoDel for the optimal management.

Goals

In [9] are explained the goals of FQ_CoDel:

- Prioritization and flow isolation for traffic.
- Benefit sparse streams and small time critical packets.
- Flow starvation prevention.

The algorithm

AQM dropping techniques may not be sufficient to satisfy strict latency requirements. In order to achieve it scheduling algorithms can isolate traffic per flow or class guaranteeing specific constraints. In detail, FQ_CoDel uses Deficit Round Robin (DRR) [26] scheduler to loop across the sub-queues.

FQ_CoDel generates sub-queues according to incoming packet flow parameters and it is based on three lists of subqueues: new, old and empty.

Initially, all subqueues are in empty state mode, but as soon as the first packet enters the buffer, the algorithm put the subqueue in a new status. Each subqueue has a quantum of bytes to dequeue packets, onces it is used, the queue will turn "old". In future sections (section 6.2), the complete algorithm will be detailed.

On the other hand, CoDel will be the AQM model to use inside the subqueues. It will apply its control law, discarding at least one packet from the head if it is needed. This avoid buffer overflow and guarantee law queueing delay.

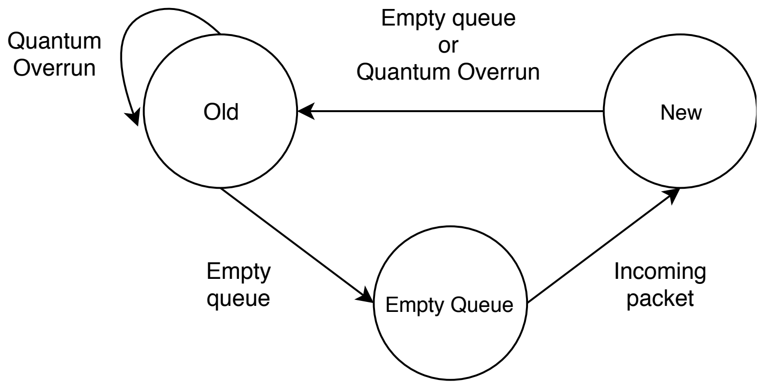


Figure 14. FQ_CoDel Queues Model

Upshot

FQ_Codel is an extension of CoDel algorithm that offers a multiple queue solution in order to provide low delay and quality to determined packets. It could be one of the solution to latency decrease in the future Internet an it is configurable in Linux 3.5 servers.

Multiple queue provides queue isolation in exchange of router process usage. However, it offers great results to latency sensitive applications. On the other hand, there are some issues to correct inside the protocol. Apart from CoDel issues, FQ_Codel has his own problems such VPN usage or wireless links with high RTT variability.

5.5 ABE

In previous analyzed models, notification of the congestion could be done via dropping or marking. Until now, dropping choice have been used (or highlighted), however, the use of marking could notably benefit the delay of the flows. Especially, enabling Explicit Congestion Notification (ECN) provides a feedback by adding new information to Internet Protocol (IP) and TCP headers. This way, packets that are going to be delivered could use their packet information to notify the sender to modify the window. Is true that ECN requirers a configured negotiation, however, according to [20] in 2014, the majority of the top million web servers provided server-side ECN negotiation.

Alternative Backoff (ABE) with ECN consist in marking packets letting individual TCP senders use a larger multiplicative decrease factor. Thus, is a sender side modification, being able to deploy it gradually pushing the use of ECN on Internet nodes.

Goals

RFC-8511 [28] establish the following goals for ABE:

- Apply different behavior with the ECN mark and packet loss.
- Being safety against the flows that do not use this TCP congestion control deployment.
- Treat the ECN signal as packet loss if the node is non-ECN-Capable.

The Algorithm

According to the ECN specification [25], the indication of ECN should be treated as a congestion loss signal in TCP that is not capable with ECN. In this case, the congestion windows will be reduced to the half and trigger slow-start mechanism. For enabling low latency and high throughput (one of AQM's main goal), [20] propose the following:

- Packet loss: TCP should reduce the congestion window $cwnd$ as usual (NewReno reduce 50% and CUBIC 30%)
- ECN mark: TCP sender reduce $cwnd$ by less than the loss.

Therefore, sender side will modify his command window size depending of the value of β, being β the decreasing factor. The appropriate value of β will drain more or less aggressively the queue, and it will be decided by the sender. Actually, two β values will exist, that one created by ECN notification and the one created because a packet loss. The paths that are not able to use ECN notification, will use $\beta = 1$, like a usual packet loss. [28] recommends the value of $\beta_{ecn} = 0.8$, being only applicable to the standard TCP congestion control.

Upshot

Alternative Backoff with ECN provides a optimization of queue management nodes. It is an additional help for decreasing latency, being, this time, the change point in the sender. The collaboration between path nodes and senders have been defended by researches like Van Jacobson, pushing the use of ECN for the notification instead dropping packets.

However, ABE has a big issue. To be effective, is decisive to have ECN capable devices in the sender, receiver and the AQM nodes between them. If the node is non-ECN capable, ABE will not be take part in the latency reduction, being the notification of congestion the traditional dup-ACKs. In spite of of not exploit ABE mechanism, it permits working between capable and non capable marking devices. Apart from that, not all β parameters are good for the rate reduction. In future parts of the thesis (section 6.3) will be explained that the recommended value is between 0.7 and 0.85. This value is set in the sender, which has to be able to configure it.

On the other hand, traditional ECN security considerations should be applied, despite of not modifying the transport protocol messages exchanged. ECN could generate unfairness situations, however, ABE is applied only when ECN-marked packets arrived, not when packets are lost. Thus, ABE could not lead to congestion collapse.

ABE is a good improvement inside AQM methods, using the sender side for it. ECN mark usage provides the advantages that are needed, improving the results of the network usage.

5.6 Decision Criteria

AQM models goals are decreasing queueing delay, minimize packet dropping and increasing throughput to the maximum possible. Is required to evaluate them in a general point of view and, also, in some important skills inside networking features. This section will show the response of the algorithms in divers scenarios that will help to decide the final AQM technique to research.

First point of comparison will be latency Cumulative Probability in online games that share the network with other type of applications (FTP, Web and VoIP). The result shown will summarize 17 types of scenarios where traffic varies from light to high. The detailed evaluation scheme is explained in [29]. Secondly, the same experiment will be done but focusing on VoIP application. After that, raw TCP usage will be displayed, showing the throughput usage in each model. Finally, the last point of comparison will be against network features such complexity of implementation, adaptability in the network, scalability and the performance in general terms.

The goal of this section is to compare the alternatives in order to select one that will be detailed and improved with some changes.

While is true that Alternative Backoff is not an explicit AQM model, it will be evaluated at the last point against other models.

5.6.1 Comparison in Online Game Latency

Inside delay sensitive applications, online games have huge impact in the network. In general, online games require low delays and low packet loss, however, there are differeces between them. Quick action games, like shooters, required the minimum delay with updates of the environment and other players position. Furthermore, cloud-games host all the environment on the servers, offering the data and voice updates in streaming to the users, in this case, the network is required to be with low delay. In [29] a maximum RTT of 80 ms is suggested, being the maximum queueing latency of 20 ms.

Overall, online games require a quick response of the network, for this, AQM models will be tested in several scenarios (low, medium and high traffic) and they will be compared in a graphic like shows the Figure 15.

Cumulative Probability shows the probability of scoring a certain or fewer packet latency, for example, Bufferbloat has a 10% of probability of scoring a 100 ms (or lower) packet latency. the graph shows the models and their scores, being Bufferbloat the traditional Drop Tail and Buffer-Control the DropTail method but with the target of scoring 50 ms buffering latency.

In review of Figure 15, both Drop Tail methods experience the worst latency cases. On the other hand, the AQM methods provide better results, being less affected by the congestion scenarios. In detail, CoDel and FQ_CoDel measure the latency directly (independently of the egress data rate) and PIE predicts the latency based on recent history of egress data rate, adapting better to the congestion.

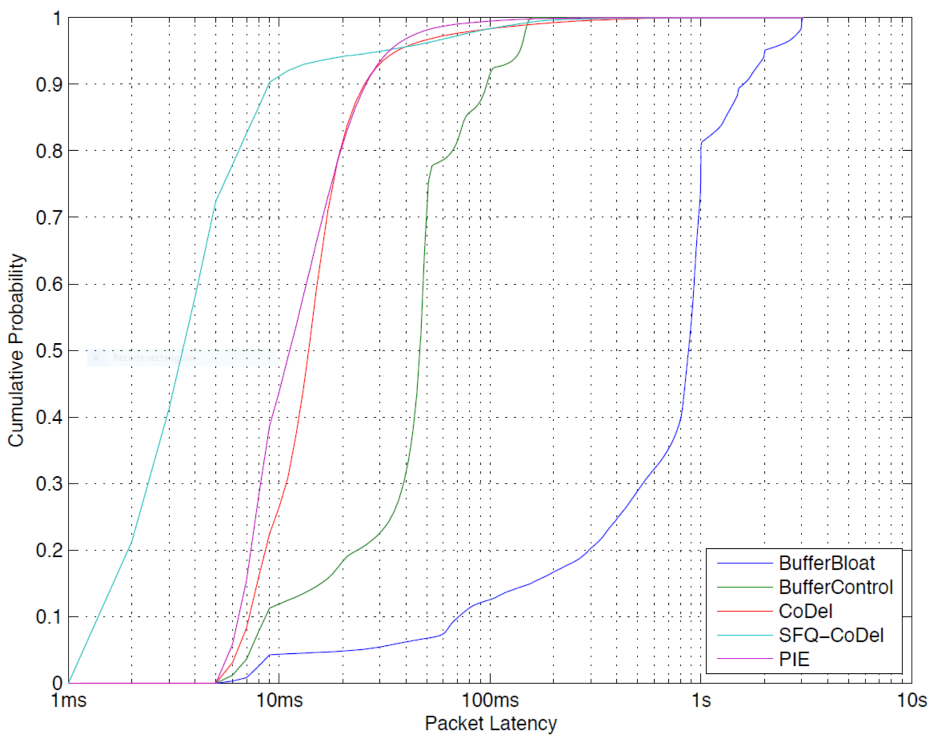


Figure 15. Gaming Packet Latency [29]

FQ_CoDel outperforms all other techniques, being the worst score in heavy traffic scenarios where could be worst than CoDel or PIE [29]. FQ_CoDel could grant good queues latency for small FTP packet cases. Though, against BitTorrent traffic (large packet flows) decrease the performance notably. Some reports suggest the DiffServ utilization in BitTorrent [4] packets in order to provide a good network resources sharing.

In short, speaking about latency in online games, FQ_CoDel provides the best performance in almost all the scenarios. As it has been shown, in summary it uses 9 ms (or less) of latency queue in the 91% of the packets.

Despite of not focusing in packet loss in this thesis, it is required some evaluation of algorithms in this aspect. Table 1 shows an overall evaluation of the algorithms. According to [29] the threshold of online games packet loss is a 20%, not being exceed by none models.

Thus, FQ_CoDel is the active queue management method that best achieve both, latency and loss, thresholds, even in high traffic situations.

Table 1. Online Game performance comparison

QUEUE MODEL	LATENCY PERFORMANCE	LOSS PERFORMANCE
Bufferbloat	Extremly Poor	Very Good
PIE	Good	Very Good
CoDel	Good	Good
FQ_CoDel	Very Good in most conditions	Very Good in most conditions

5.6.2 Comparison in VoIP Latency

VoIP applications demand have been increasing last decade, requiring low delay in the transmission of the packets. These packets are small and it is permitted a certain loss of them, thus, the main concern of VoIP is latency. For it, the quality of VoIP wil be illustrated with Mean Option Score results (MOF), being the ideal score 4.4 of 5.

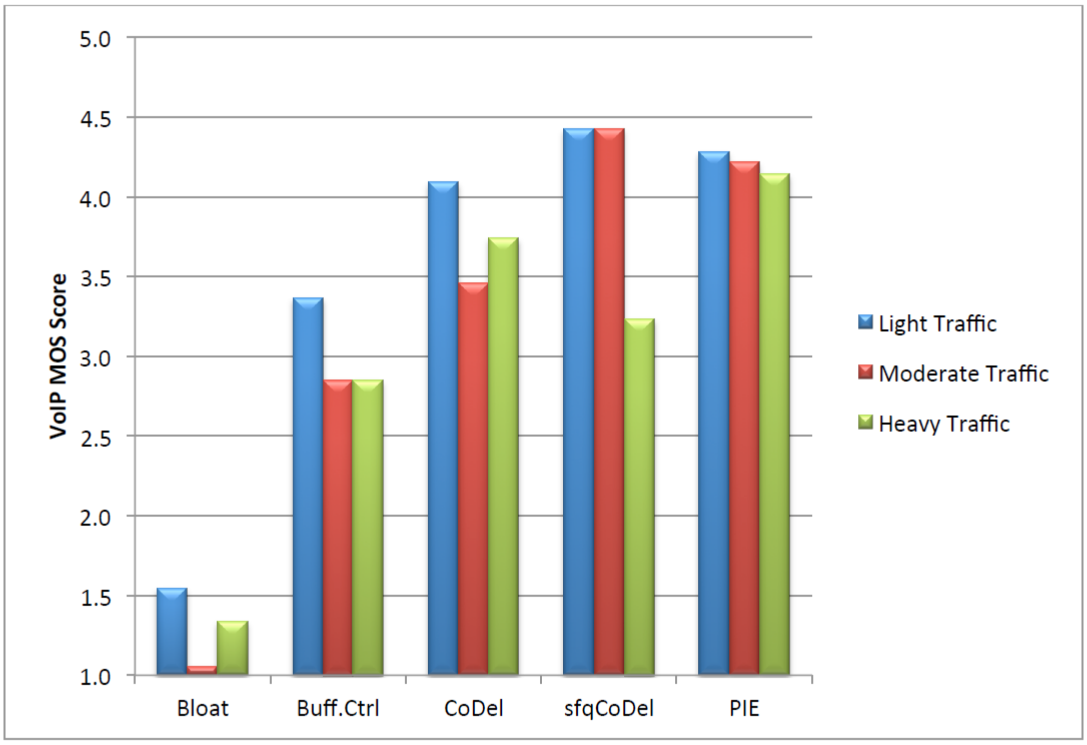


Figure 16. MOF in VoIP [29]

On the whole, PIE algorithm provided in all scenarios near ideal MOS score, being followed by FQ_CoDel that is really good in light and moderate traffic cases. Finally, DropTail methods do not are in large difference with the AQM methods.

5.6.3 Comparison in TCP bulk upload

This subsection will treat the performance of TCP-Reno upload without competing trafic. Thus, the throughput treatment will be evaluated, being the topology a bottleneck that cross from 20 Mbps to 5 Mbps link. Drops of AQM methods and, consequently, the throughput use will be the point of this part.

In this case, PIE has the worst performance comparing with other AQM methods. Obviously, bare Drop Tail mechanism has the best result, due to the maximization of TCP throughput usage and Bufferbloat. Drop Tail generate a stable 5 Mbps throughput all the time, however, AQM methods are more unstable.

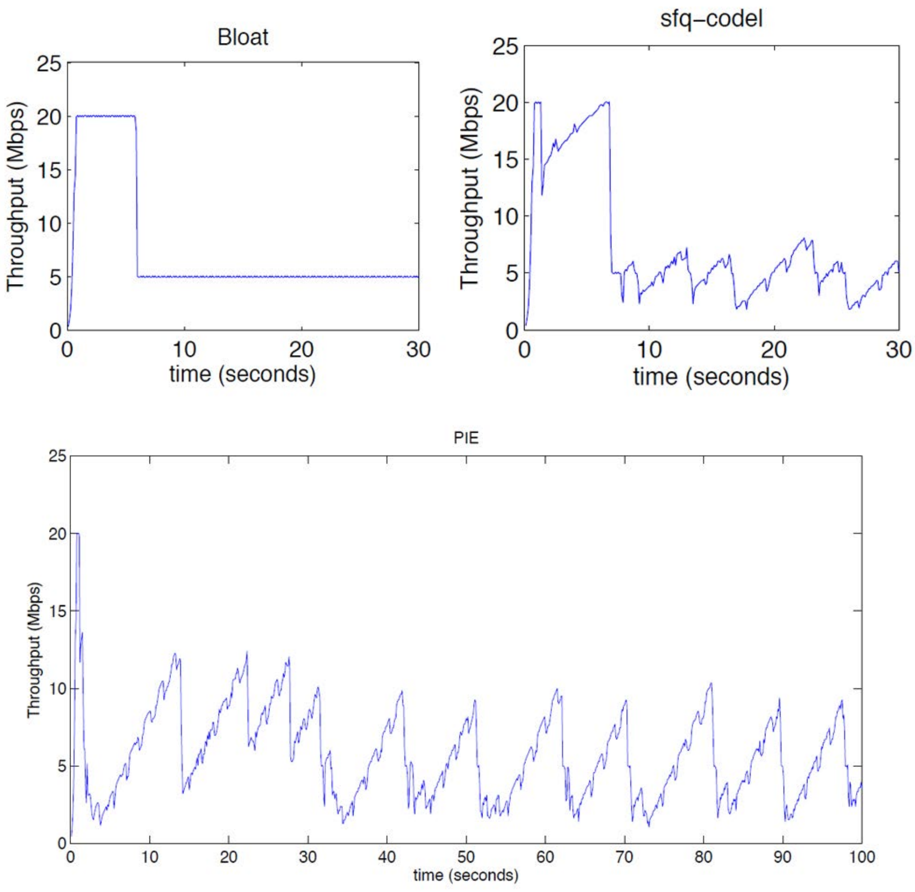


Figure 17. TCP Bulk Performance [29]

The Figure 17 do not show CoDel performance. This is because FQ_CoDel and CoDel have the same performance in one queue (no traffic and only one TCP stream) situation. They have enough buffer to allow the increase to 20 Mbps, however, it tries to react to the increase of the congestion window and it starts dropping packets to achieve the minimal standing queue. Thus, the throughput decreases notably until 5 Mbps and then it oscillates around this number. Therefore, in a simple TCP usage, without traffic, the link usage is not maximized.

The usage of the link, throughput, could be increase with the use of ABE, changing the value of the reduction factor in TCP sender. Later, in section 6.3 this procedure will be explained in detail.

PIE method perform worst than the others. The ramp to 20 Mbps is immediately decreased to 1 Mbps and then it varies between 12 and 1 Mbps more heavily than in the others AQM techniques.

5.6.4 Comparison in general features

Developing new techniques for network improvement is a challenging task. These techniques should be simple to implant, adaptable to changes, scalable for new network environment and with the optimal performance. Thus, it is required a general features analysis of the active queue management models.

	Complexity	Adaptability	Scalability	Performance	TOTAL
RED	3	1	2	1	7
CoDel	2	3	3	2	10
PIE	2	3	2	2	9
FQ_CoDel	1	3	3	3	10
ABE	1	2	2	2	7

Table 2. General Features Table

Table 2 gives marks to the AQM models in base to their skills. After explaining them and testing the performance, the total score is showed. The algorithms have been punctuated in a 0 to 3 range, being 3 the best option and 1 the worst. The total score is the sum of the row.

Complexity

Complexity of implementation evaluates how hard is to deploy the technique in nowadays network. RED is the easiest one to install, almost all the routers permit this method and turning on is really easy. On the other hand, FQ_CoDel is one of latest modern technique proposed, requiring equipment with enough resource power capable of managing around 1024 queues. However, almost all Internet equipment is ready to deploy it. Apart from that, ABE's implementations is hard due to the use of the ECN mark. For scoring the optimal usage of ABE, it requires that all the nodes in the path are ECN capable, being not easy to achieve it. In addition, the sender should be configured, usually they are customers equipment, being necessary the configuration on the ISP side.

Adaptability

Adaptability is an important point to evaluate, it is defined as the capability of the model to adjust to traffic changes. Node's egress link is shared by multiple connections, in different times and duration. Therefore, queues will change relying the number and type of connection at that time.

RED "tunning" parameters generates a hard adaptability against traffic changes. Network administrators need to set the ideal parameters, varying in time and in the scenario, becoming the worst model at this point. On the other hand, AQM techniques like CoDel, PIE or FQ_CoDel have a really positive response to network changes due to being parameterless.

Scalability

Scalability is defined as the capability of the model to handle with work or accommodate to an increase of work. Network is constantly evolving, with new applications an lower latency requirements, beside to permit to work as well with old flows. In this case, almost all the methods have good scalability scores, being CoDel and FQ_CoDel the best ones in this field. Despite of requiring good computational skills, FQ_CoDel will be able to work with future flows, quick and with the necessary quality of service.

Performance

Performance point has being analyzed in previous sections. In summary, in spite of not having the best response with high amount of traffic scenarios, FQ_CoDel has te best results. RED, though, could not offer the requirements of latency and throughput for nowadays's time sensitive applications.

5.6.5 Final Deicision

In accordance with the evaluation sections, FQ_CoDel and CoDel have the best features and scores. In fact, as it is shown in Table 2 they are in tie in Total points. CoDel is easy to deploy and it offers an stable and good response to traffic scenarios. It does not require extra configuration in path nodes and has the capability of using ECN marks. On the other side, FQ_CoDel grants better performance than CoDel but being more laborious the deployment. However, it scores the best performance, being capable of providing flows independence and ultra low queuing latency to the packets. Besides having a throughput performance comparable with other AQM methods, it offers queue delays lower than 9 ms. In addition, it is ECN capable and most of the researches defend it as the future queue management system.

ABE has been included in the thesis as a queue latency manager, however it is a sender side technique that offers the opportunity of treating ECN marks in a different way than the dropping. By itself, does not provide the best performance, but it could be a complement to the evaluated AQM models. In fact, the thesis proposal will be the combination between FQ_CoDel and ABE.

The merge between FQ_CoDel and Alternative Backoff with ECN could provide the notably reduction of latency. FQ_CoDel is located in the router queue, while ABE is in the sender, working together they could manage the packet transmission optimally and grant a quality of service and experience for low latency application users. In addition, it could encourage the use of ECN in the nodes which nowadays is not widely used.

The use of AQM algorithms with ECN have been defended by many researchers, therefore, it could be the path that provides the buffer latencies that new network applications will require.

6 Proposed solution

The proposed solution will be explained staring by the components that create the set of FQ_CoDel + ABE solution. First, CoDel will be explained in detail, understanding the algorithm and the behavior. Once CoDel is explained, the next subsection will spread the behavior to more than one queue (FQ_CoDel), analyzing the working mode and the decisions that the algorithm do. Finally, ABE will be defined, presenting the combination of it with FQ_CoDel algorithm.

Once all the models are analyzed in detail, the simulations of the algorithm's behavior will be displayed, with all the scenarios and the scored values. The results are shown in the final section, with all the values an evaluation against the other algorithms explained in the analysis of alternatives (section 5).

6.1 CoDel in detail

RED algorithm's effectiveness depends on the correct tune of four parameters ($min_{threshold}$, $max_{threshold}$, w_q, max_p) whose varies depending on the network situation. Setting the appropriate values is a critical issue which limits the implementation of the algorithm on the actual Internet network. Controlled Delay (CoDel) [14] avoids the use of "tuneable" parameters used to predict the congestion packet sojourn time, instead of average queue size. Additionally, it is heavily adaptable to several network scenarios due to the independence of parameters such as queue size, queue size average, drop rate, link utilization, among others. All these features make CoDel one of the most valuable algorithm inside the active queue management solutions.

The main point of CoDel is that relies on the sojourn time of the packets for the queue management, in other words, queue control is based on the measure of the amount of time that a packet spends inside the queue, being the start time when the packets enter the node and the finish time the instant when the packet is forwarded or dropped. Besides that, CoDel has five particular characteristics in contrast to other techniques:

1. The congestion ("bad" queue) will be tracked with the local queue minimum.
2. The static value for tracking will be a single state variable.
3. The observed datum for each packet will be the sojourn time.
4. The network power will be maximized with the optimal setpoint.
5. Modern state-space controller will be used.

Next sections describe CoDel algorithm in detail, however, first, is necessary to understand two crucial concepts [14].

- **Sojourn:** Is the difference between the departure time and the arrival time of a packet inside the buffer.
- **Standing queue:** Is the queue that persists more than the longes round-time-trip.

6.1.1 Building Blocks: "Bad" and "Good" queues

As explained in the analysis of alternatives, one goal of CoDel is to distinguish between "good" and "bad" queues in order to get rid of "bad" queues. The usual problem of TCP is that the sender focus in reducing the window size without having in account the packets in flight that could process the path. On one hand, bottlenecks that do not afford sender's windows rate, producing a queue that stands more than one RTT, are known as "bad" queues. On the other hand, queues that process the packets (including packet burst) in less than one round-time trip are "good" queues. Thus, is necessary to detect "bad" queues and control them.

CoDel's Request for Comments (RFC8289) document [14] explains the differences between the bad and good queues using the following example. Let‘s imagine that a sender has a packet window of 25 packets through a 20 packets of BDP connection. At the first burst, 5 packets will be put in the standing queue (because of the mismatch between the window and pipe size). In one RTT 20 packets would be delivered, but 25 packets would be in flight. These 5 packet standing queue generates continuously delay to the path, hence this queue is a "bad" queue. In case of "good" queues (20 packets of *cwnd*) it would go away in one RTT.

6.1.2 Queue control components

CoDel algorithm could be divided into three components: the estimator, the target setpoint and the control loop. Estimator figure out the situation of the queue, target setpoint would establish what the queue needs and the control loop will take actions to move to the point that queue needs. Now they will be detailed.

Estimator

The estimator is the component that deals with the detection from "good" to "bad" queue turning. Queue length is the usual AQM method's indicator, however, it is difficult to adapt it quickly to changes, further in scenarios like Wifi where the link changes are longs and shorts. Apart from that, sometimes, dropping packets generates spikes that should be ignored by the observer, due to spikes, queue length increase, but they do not persist. The monitor should be focus in the amount of excess delay that a packet experience due to a persistent queue, in other words, sojourn time.

AQM techniques are focused on decreasing the queue latency, therefore, it is reasonable to use sojourn time as the tracking variable. It provides several advantages:

- It is independent of the link rate.
- It generates better performance than using queue length.
- It is directly user-visible.
- It works with rate changes or link sharing situations.
- It is independent of the number of queues or the service type.

Once the sojourn time is established as monitor value, it is required to separate "good" from "bad" queues. Normally, the average is used as an indicator, being the average queue one half of burst size, however, it depends on the time when the average is computed or the time of arrival. CoDel proposed the minimum sojourn time as an indicator, being a better option. If only one packet is in the queue, the sojourn is zero, no persistence.

Thus, the **indicator** of CoDel algorithm is the (local) minimum sojourn time.

Another point to determining is the **interval** of time of updating the indicator. It should be between the maximum time where the minimum delay is in effect and the time to drain the largest burst. According to [14] the correct value is the maximum RTT of all connections.

In summary, the estimator building block uses the sojourn time as the observed value and the local minimum sojourn as the static to monitor. It provides a robust and accurate measure of the queue, only decreasing when the packet is dequeued, put it differently, CoDel only mark in the dequeue instant, avoiding packet locks.

Target Setpoint

At this point it is known the value for monitoring the congestion, nevertheless, it requires to set a point that tells when to act. If that point generates an early act of CoDel´s dropping when the estimator is not zero, the drop rate will increase, but the goodput will be seriously harmed. Hence, the goal is the point that maximizes the utilization while minimizing the delay. For it, as it is explained in [14], power will be the value that measures the relationship between the nominal throughput and the nominal delay.

The RFC8289, based on power values graphics, determines that the setpoint target should be the 5% of the nominal RTT. That value had been evaluated in Reno, Cubic, and Westwood scenarios and it works properly. So, considering 100 ms of RTT in a terrestrial value, the target setpoint should be 5 ms. That is to say, when local minimum sojourn time is above the target setpoint (5 ms in terrestrial Internet), CoDel will start acting.

Control Loop

Working with control theory in queues is a hard job because they are not a linear system, and AQM operates at a maximum of non-linearity. In addition, due to the mix of flows, they are time-invariant and they depend on the protocol used for the connections (TCP or UDP i.e.). Even so, classic controllers behavior is not compatible with determining the persistence of a queue.

With all, is necessary control that drives persistent queues that are above the target to queues below the target. The solution consists of dropping (marking) packets, however, dropping in a high rate generates a bad effect in the throughput. Thus, an appropriate period of time between dropping should be determined. According to [14], the best method is to start at a low rate of dropping and increase slowly until the persistent queue goes below the target. This way, the over dropping is avoided and is guaranteed to eventually dissipate the persistent queue (stochastic gradient procedure).

The time between drops should be set conveniently, the controller must detect it before the next drop, otherwise, CoDel will overdrop. The minimum time that controller needs to see the effect is the round-trip time plus the local queue waiting time. Though, the local waiting time tends to vary so the appropriate time will be a little larger than the RTT, being a convenient measure the estimator interval. Interval will ensure that the controller has accounted for acceptable congestion delay, avoiding overdrops.

Apart from that, the time interval should be decreasing slowly in order to exit from the persistent queue. For it, the interval time will be decreased according a dropping counters since the dropping period starts. In essence, the next drop time will follow the equation 6.1

$$NextDropTime = \frac{INTERVAL}{\sqrt{drop\_count}} \tag{6.1}$$

6.1.3 Non-Starvation

CoDel needs to know when it would starve the output link in order to stop dropping packets. For it, it would check if at least an MTU worth of bytes remains in the buffer. If not, the packet will not be dropped and CoDel will exit dropping state. MTU size will be the largest packet that it has seen.

6.1.4 Overview

CoDel is one of the AQM methods that works against bufferbloat, having really good performance with low delay and with a simple implementation. It only requires to set up the INTERVAL value, 100 ms for Internet usage. On the other hand, TARGET looks for optimizing the power value (maximum link utilization with minimum delay), being the ideal value the 5-10% of the connection RTT, so, for the Internet TARGET would be 5 ms.

In summary, CoDel's procedure is the following one: when estimator finds a delay above TARGET, the controller enters in a drop state and it sets the next drop time, being the first drop time equal to INTERVAL value. After it, the controller will calculate next dropping time according to the number of packets that it has dropped, until the delay decreases below the TARGET. CoDel will use the prevention mechanism for entering too early to drop state after exiting it. Algorithm's diagram, as has shown in previous section 5, is the following Figure 18:

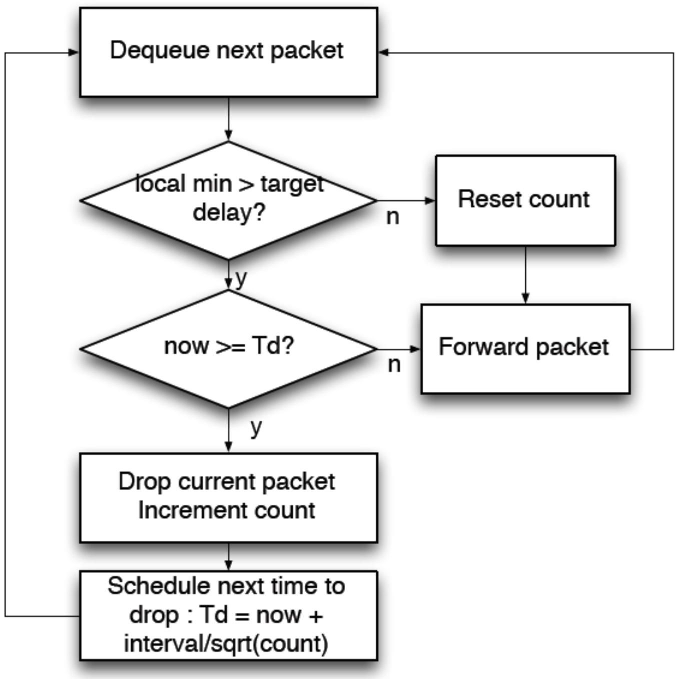

Dequeue next packet
local min > target delay?
n
Reset count
y
now >= Td?
n
Forward packet
y
Drop current packet Increment count
Schedule next time to drop : Td = now + interval/sqrt(count)

Figure 18. CoDel Algorithm Simplified

6.2 FQ_CoDel in detail

CoDel's behavior grants the opportunity of extending it to more than one queue. The independence of the variables and the capacity of adapting to changes makes the researchers evolve it to FQ_CoDel algorithm.

FQ_CoDel is the algorithm able to manage more than one queue, applying CoDel in each one. It works with multiple flows, decreasing the impact of bufferbloat and providing flow isolation for low-rate traffic. Actually, the queue management of FQ_CoDel is thanks to the combination of Deficit Round Robin (DRR) and CoDel, being able to implement it in some network simulators software and in the Linux kernel.

Before explaining in detail the algorithm, some terminology needs to be described:

- **Flow:** Is defined as the group of packets that share the same source IP address, destination IP address, source port number, destination port number and protocol number (5-tuple of IP). FQ_CoDel hashes the flows in different queues for managing them,
- **Quantum:** The maximum amount of bytes to be dequeued from a queue at once. By default is set to 1514 bytes, the Ethernet MTU plus the hardware header length of 14 bytes [9]

The explanation of FQ_CoDel will be split in: flow queuing management and the scheduler. The algorithm stochastically classifies incoming packets in different queues, then, the scheduler selects the queue to dequeue packets from.

6.2.1 Flow Queuing

Each flow that arrives at the node needs their own queue, being the "flow queueing" the term to define this behavior. By default, 1024 queues are created, assigning one hash to each one. That hash is created by the combination of a random number (selected at the initialization of the algorithm) with the 5-tuple hash (IP destination, IP origin, destination port, origin port, and protocol). Inside each queue, packets are in FIFO order and the AQM model implemented is CoDel.

FQ_CoDel selects the queue to apply the AQM by the deficit round robin mechanism, bytes-based. Each queue has a number of bytes-credits initialized in a "quantum" variable (the granularity of dequeue). Once the queue has dequeue chance, it decreases the number of credits by the packet size of each packet, and it will be doing it until the value of credits becomes zero. After that, the pointer is increased by one quantum and the dequeue opportunity is finished.

With this procedure, small packets are not penalized by the difference in packet sizes, DRR applies byte-based fairness. In detail, DRR acts in two queues sets: "new" and "old".

6.2.2 Scheduler

Once the packet arrives at the node, and the packet has not the previous bucket of the queue, it is added to a new queue and it becomes active, here starts the **enqueue** time. After a time, is moved to old queues, being removed if it is not activated. In fact, the enqueue procedure is divided into three stages: classification, timestamps, and the optional dropping.

When a packet arrives, it is necessary to **classified** into a bucket of the queue. For it, the packet's hash of the 5-tuple IP with value modulo the number of queues is calculated. The hash is salted with a random number (selected at the initialization time) to prevent DoS attacks. At this time, the packet is classified into a queue and starts the time of CoDel and the **timestamping**. The packet, following a FIFO order, will be pointed at the tail of the selected queue, and the queue will be positioned at the end of new queues, with the quantum number of credits. In addition, the queue‘s byte count will be updated by the packet size.

Each FIFO queue is tracked by two state variables, queue credits and the total number of bytes enqueued, apart form CoDel state variables. Additionally, other variables could be included, for example, Linux keeps counting the number of packets dropped. The number of total bytes enqueue is used for the limit on the number of packets in all queues. The default packet limit is 10240, suitable for up to 10-Gigabit Ethernet speeds.

The system has an overload protection mechanism. In the case of the total number of enqueued packets exceed the limit, the queue with the largest byte count is selected and half of the number of packets is dropped. Using the byte count for selecting the queue, decreased the CPU usage because of is not necessary to find the longest queue.

The **dequeue** time is when FQ_CoDel works the most. First, selects the queue to dequeue, then dequeue it and, finally, it process some values.

FQ_CoDel starts reading the list of "news" queues in order to **select** the dequeuing queue. It starts from head to the bottom, and if the queue has the number of negative credit, it is given an additional quantum of credits and moved to the bottom of old queues. If the queue has a positive number of credits, that will be de one to dequeue.

If the list of new queues is empty, the scheduler will start the same procedure in the old queues list.

Once a queue is selected, CoDel is triggered. If packets need to be dropped or marked, CoDel will select those from the head of the queue and then the selected packets will be **dequeued**.

If no packet is returned to dequeue, the queue must be empty, thus:

- If is a new queue, will be moved to the old list.
- If is an old queue, the queue will be removed and added to empty queues list (until the new packet arrives).

The dequeue process consists of the return of a packet from the CoDel mechanism to the scheduler. When it happens, the schedulers subtracts the size of the packet to the queue credits.

Finally, moving empty new queues to old queues list prevents the starvation. The empty new queue could reappear (with the arrival of a new packet) before old queues are visited; thus, it forces to serve all old queues before the empty queue is removed and avoiding the starvation.

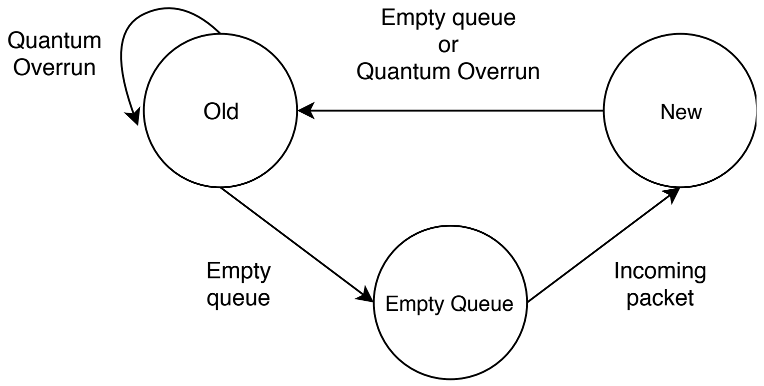


Figure 19. FQ_CoDel Queues Model

6.2.3 Differences from CoDel

As have been explained, each queue maintains the state variables for FQ_Codel, therefore, if only one queue is active, FQ_Codel acts identically as CoDel. However, some differences exist between both two algorithms.

Finding the convergence point of dropping (the one that minimizes the delay and maximizes the throughput), requires a certain time for the algorithms. FQ_CoDel's schedule requires to perform the flows-competing and the treatment of old and new queues. Thus, it takes more time to converge to the ideal dropping rate.

Conversely, FQ_CoDel flow isolation provides the drop of packets more accurately from the largest flows than CoDel (only one queue). In addition, it provides better times against traffic behavior changes, reacting almost immediately.

6.3 ABE: Alternative Backoff with ECN

The congestion control signal is noticed by backing-off a packet loss which occurs, in loss-based TCP algorithms, when the bottleneck's buffer is full. AQM provides great results for fighting the latency increase, but usually, it uses a drop of packets instead of marking them. Marking packets with the Explicit Congestion Notification (ECN), delivers congestion feedback via packets and avoids the packet drop in congestion situations.

The rule of thumb, or BDP law (explained in section 4), grants the 100% of throughput usage, however, delay values are notably increased due to the transport of the double of packets in flight through the path.TCP suffers therefore when the path's BDP excess the bottleneck AQM schemes' effective buffer size, and starts signaling the congestion.

Alternative Backoff with ECN (ABE), proposes the use of ECN marking with the combination of AQM methods that enables low latency and high throughput even for high BDPs. AQM modes have performance degradation with scenarios with high RTT (above 150 ms), usually in inter-continental uses. Thus, ABE tries to increase the performance with a sender-side modification that can be deployed incrementally with the ECN usage and being able to work with non-ECN capable flows.

The proposal of ABE consists in the modification of the reduction of congestion window depending on the notification type, in summary:

- Packet loss: TCP will reduce the *cwnd* as usual (50% in Reno, 30% in Cubic).
- ECN mark: Sender will reduce the *cwnd* less than the usual response.

6.3.1 Explicit Congestion Notification (ECN) situation

In 2001, RFC3168 [25], proposed the use of ECN as a simple marking method for AQM models, avoiding dropping packets. It was deployed and implemented in host and nodes, however, it is not widely used. As it is explained in [20] the majority of webservers in 2014 have ECN negotiation and it is supported by the newest AQM algorithms.

6.3.2 AIMD problem

AIMD or Additive Increase, Multiplicative Decrease (AIMD) force the *cwnd* reduction to $\beta = 0.5$, due to the AIMD multiplicative working style. However, this backoff value does not use optimally the capacity of the link, as it is shown in Figure 20.

On the other hand, the use of AQM with the TCP change behavior provides better performance of the link as it is explained in Figure 21.

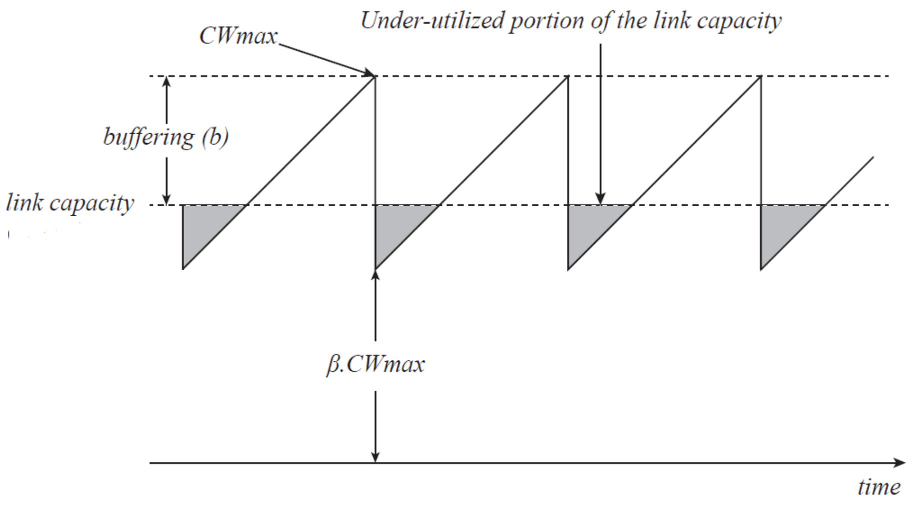


Figure 20. TCP Sawtooth Problem.

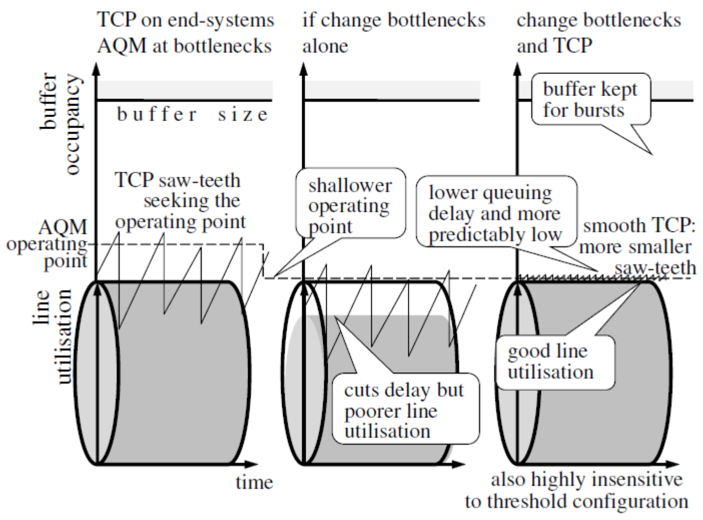

TCP on end-systems AQM at bottlenecks
if change bottlenecks alone
change bottlenecks and TCP
buffer occupancy
buffer size
buffer kept for bursts
TCP saw-teeth seeking the operating point
shallower operating point
lower queuing delay and more predictably low
smooth TCP: more smaller saw-teeth
AQM operating point
line utilisation
good line utilisation
cuts delay but poorer line utilisation
time
also highly insensitive to threshold configuration

Figure 21. AQM plus ECN Proposal.

6.3.3 Solution

ABE generates a new backoff value for TCP senders, $\beta_{ecn} > \beta$ in response to ECN marks, but maintaining the loss remains. It goes against the ECN official recommendation of treating ECN marks as same as a congestion notification, however, it is probed that ABE increases the optimal usage of the latency. AQM behavior, therefore, assumes that an ECN mark is created by an AQM model, having queue congestion.

Selecting β_{ecn} value

There is a point of discussion in [20] which determine the value of β_{ecn} for the propped usage of ECN marking. Taking into account that, as a larger value of β, higher the throughput and latency, is crucial to select a value that maximizes the throughput and minimize the latency. Thus, it consists of searching the balance between the utilization and bottleneck draining rate. It does not exist the perfect value for all the scenarios, nevertheless, [20] selects a value between 0.7 and 0.85 as the ideal one, especially after the slow start period. The next graphic showed in Figure 22 and taken from [20], where the value of β is discussed and evaluated.

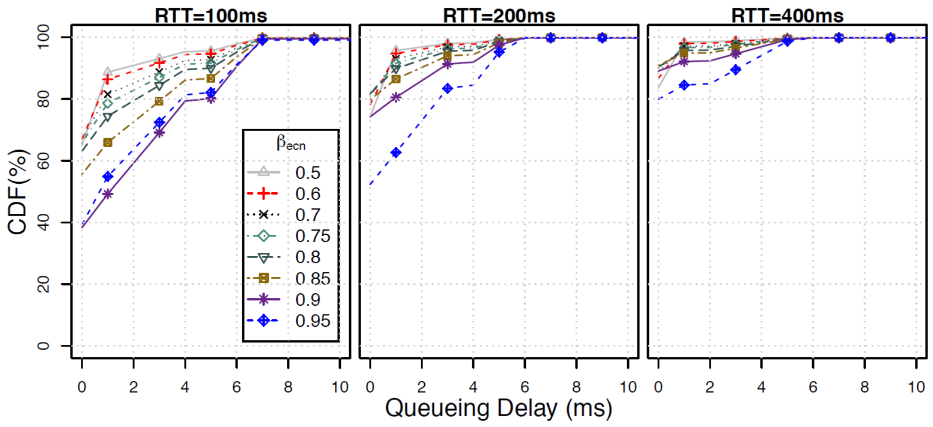


Figure 22. CDF of Queueing Delay in TCP Reno and AQM CoDel (10 Mbps).

As it is shown, CoDel keeps queuing delay close to the default TARGET value of 5 ms for all ranges of β_{ecn}. In detail, the values between 0.7 and 0.85 are the ones that would be the optimal value for all the scenarios, getting the balance between throughput and latency.

6.3.4 Overall

ECN backoff provides an alternative to the classical TCP behavior, which allows the coexistence of both. If the path does not support ECN marking, it would be treated as the traditional TCP behavior. Researches have demonstrated, as have been shown, the benefit of ECN marking reducing the loss-based TCP congestion control interaction. One of the strongest points of ABE is that the modification is in the sender side, consequently, it does not require changes from the TCP receiver or network nodes.

6.4 Simulations

This section will explain the simulations done by vía ns3 software. Network Simulator 3 [6] is *"a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use"*. All the simulation have been developed with ns3, extracting the results in .pcap and text files. The simulated scenarios run from different congestions situations and in diverse round trip time values in order to check the algorithms responses in several examples. On the other hand, FQ_CoDel + ABE proposal have been evaluated according to the results of the official papers, analyzing their behavior and concluding the expected outcome.

6.4.1 Topology

The topology selected for the simulation is a simple dumbbell topology that generates traffic from left side nodes to the right ones. Figures 23 shows the design of it with the IP addresses and throughput values.

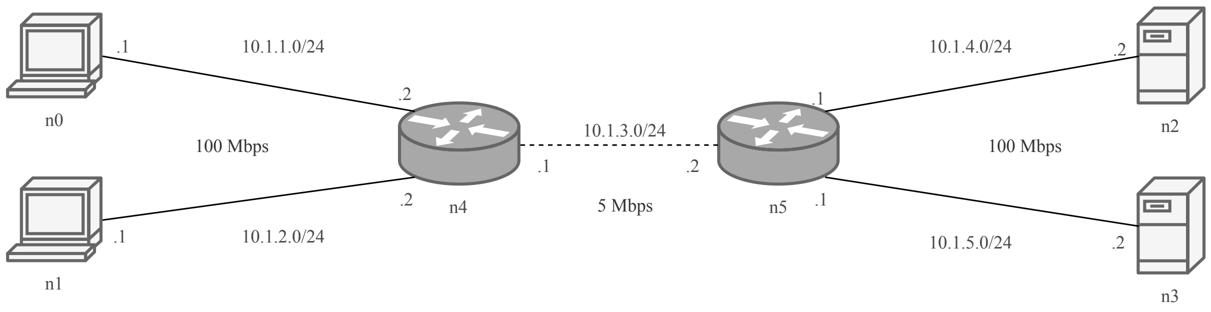


Figure 23. Dumbbell Simulation Topology

Topology is composed of 5 networks divided by two gateway routers. Edge links have a capacity of 100 Mbps, while the connection between routers has 5 Mbps capacity. Actually, the n4 router is the bottleneck of the topology, where the AQM model will be installed. The minimum buffer capacity is defined by the rule of thumb, which depends on each scenario, anyhow, $2 * BDP$ value will be used for the simulations.

Traffic will be created from n0 and n1 nodes to n2 and n3 nodes respectively. n0 will create a TCP connection with large flows, conversely, n1 generates UDP light flow connections, such as time-sensitive applications that require low latency values. Throughput values of each node will vary depending on the scenario, generating more or less congestion in the bottleneck.

Finally, all nodes, excluding n4, will have DropTail queue management installed by default. On the other hand, n4 will compare Droptail, CoDel and FQ_Codel, and FQ_CoDel + ABE for each scenario. The value of the INTERVAL and TARGET are the default ones (5 ms and 100 ms respectively), on ABE β_{ecn} value will be set to 0.7, following the [20] recommendation.

6.4.2 Scenarios

Dumbbell topology will host diverse scenarios whose will evaluate the performance of the algorithms. Changes would be done in the congestion levels and the round trip time (RTT) values. This way would be easier to see the differences, simulating real Internet situations in accurate mode. In detail, the scenarios to develop are:

- No congestion
  - Small round trip time value (RTT = 50 ms)
  - Medium round trip time value (RTT = 100 ms)
  - Large round trip time value (RTT = 400 ms)
- TCP (medium) congestion
  - Small round trip time value (RTT = 50 ms)
  - Medium round trip time value (RTT = 100 ms)
  - Large round trip time value (RTT = 400 ms)
- TCP and UDP (large) congestion
  - Small round trip time value (RTT = 50 ms)
  - Medium round trip time value (RTT = 100 ms)
  - Large round trip time value (RTT = 400 ms)

The RTT times have been selected according to three situations. Continental usual round trip time is about 50 ms, then, 100ms of round trip time is the common time on the Internet, finally, 400 ms of RTT responds to large intercontinental connections (and some satellite connections). By this way, congestion control algorithms are tested for every situation.

Speaking about congestion, three types of congestion have been evaluated. The first case is with a unique TCP connection without exceeding the bottleneck bandwidth. In detail, the TCP connection generates a flow of packets from n0 to n2 with a 3 Mbps throughput. The simulation would show the maximized throughput with the expected RTT of 100 ms, approximately.

The second case generates a unique TCP flow, but the throughput will go beyond the bottleneck capacity. In particular, 6 Mbps throughput will be installed. At this situation, AQM algorithms will decrease the maximum throughput but the RTT will be under the minimum values.

At last, two flows are created, one large TCP flow (throughput of 4 Mbps) and the second one a little UDP flow (throughput of 4 Mbps). UDP flow will be running only for 10 seconds of the simulation, forcing the queue management algorithm to adapt to traffic changes.

All the simulations span for 30 seconds, with the last 3 seconds without flows traffic. Thus, TCP traffic will be for 27 seconds working and UDP will interact from the 10 to 20 seconds interval. The subsection details the values that we want to obtain from the simulations.

6.4.3 Test's feedback

At this moment, is time to decide the graphics to present and compare the distinct algorithms results. The goal of the thesis is to find the algorithm that maximizes the throughput with the lowest latency possible, for it, throughput, round trip time, packets in queue and sojourn time will be evaluated.

Packets round trip time and throughput would be displayed via Wireshark [7], tracing the .pcap of the bottleneck flows. It displays the situation of the link every moment, showing the congestion situation and the throughput values. On the other hand, RTT shows the time that a packet needs for going to the destination and coming back. The path times (RTT) will be different in each scenario, 50, 100 and 400 ms; however, the queue times could be calculated with the difference between the theoretical path RTT and the real one measured in the pcap.

ns3 will traceback the packets in the queue every moment, showing the changes during the congestion and the queue situation for each algorithm. Sojourn time of the packets is really helpful to see how the CoDel algorithm works (TARGET = 5 ms), triggering the algorithm when the sojourn time is above the TARGET.

Finally, via FlowMonitor function, a summary of the simulation will be displayed. With the total packet dropped (before and after the queue), the throughput average, among other information. Next subsection explains all the results, comparing the algorithms and evaluating them.

6.5 Results

This section will display the results obtained from the simulations explained before. For each scenario, the main values are analyzed, emphasizing the discrepancies between the algorithms for each point. Finally, the overall result is shown, with the evaluation of the proposed algorithm.

All the plots displayed are generated with Gnuplot [5] software or Wireshark traffic analyzer.

6.5.1 No congestion

Scenario 1: RTT = 50 ms

The results here below, are the ones that show the scenario at the ideal situation: no congestion and low RTT. As has been explained in the theoretical explanation of the procedure of the algorithm, with no congestion algorithms are not triggered, thus the values of the throughput and round trip time are the maximum ones.

For 50 ms round trip time the topology is divided in a specified way: first 5 ms of delay in 100 Mbps path, second 15 ms of 5 Mbps bottleneck and, finally, 5 ms of delay in 100 Mbps path to the servers. Therefore, one RTT (50 ms) is twice the time for going from the client to the server (25 ms). On the other hand, the generated flow is 3Mbps TCP flow (lower than the bottleneck capacity), therefore there is no congestion and, regardless the algorithm installed, all the packets should be delivered without being dropped. Another characteristic of the Scenario is that bottleneck node (n4) has the double of the buffer's size of the rule of thumb, being 42 packets with 1500 bytes per packet (usual TCP packet size).

Table 3 shows the values taken from the first scenario simulation. All the values are the same for all the algorithms.

Table 3. Scenario 1 Results Values

Algorithm	Avg. Throughput	Avg. Goodput	Mean Delay	Mean Jitter	Drop Packets
All	3.29 Mbps	2.6 Mbps	28.51 ms	1.14 ms	0

The TCP flow generates a 3 Mbps traffic, being the all throughput higher due to control packets. However, the average goodput is lower because of packets headers. On the other hand, mean delay of the unique traffic, without congestion, is 28.51ms, being the total RTT of 57ms that, apart from some others delays, matches with the established RTT time. Finally, as the congestion situation has not given, there are no dropped packets.

Figures 24 and 25 show the throughput and round trip time of the TCP flow during the simulation time. As clearly can be seen in Figure 24, the throughput is maximized to the selected flow throughput value (3 Mbps), being stable after the slow start period. This is the natural TCP Reno behavior, starting from zero, increasing heavily the throughput value until arriving at the top. About the RTT, the flow maintains a 52 ms value, near the RTT established. This simple scenario shows the TCP usage under no congestion, with the slow start mechanism and the default behavior of the transport protocol.

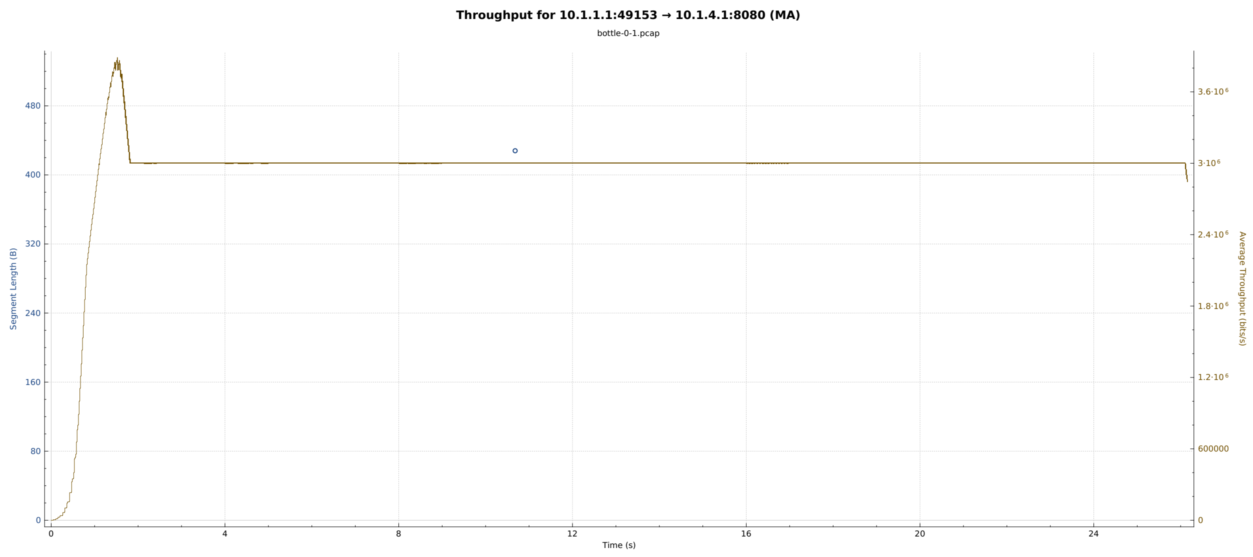


Figure 24. Throughput Drop Tail Scenario 1

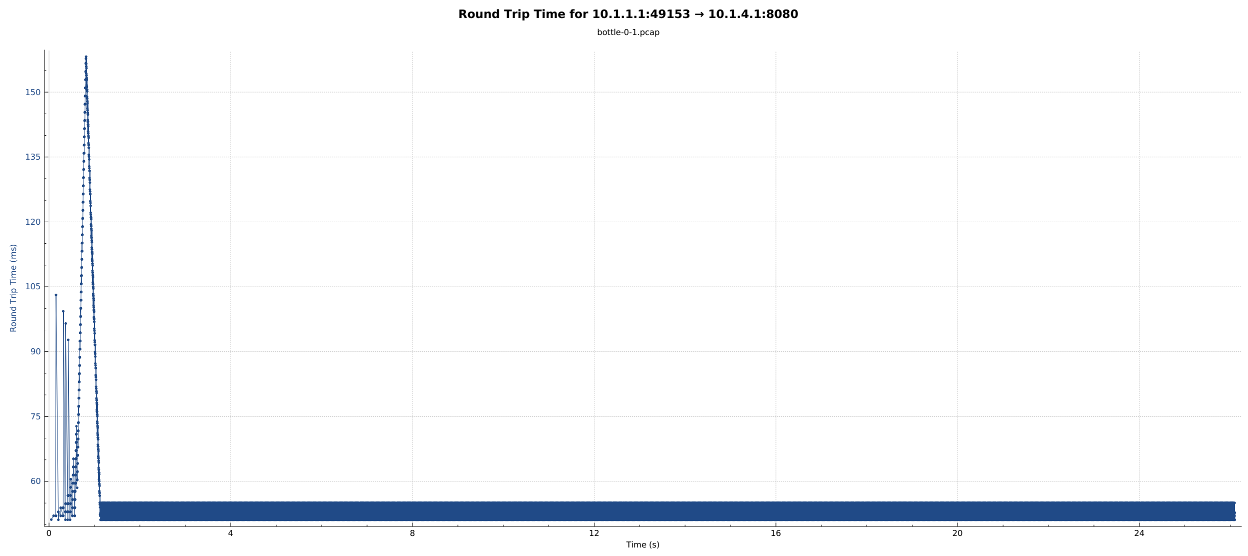


Figure 25. RTT Drop Tail Scenario 4

The presented scenario exposes the case of low RTT without congestion existence. It gets the standard values of the topology, offering a quick look to the channel's performance upon the traffic.

Scenario 2: RTT = 100 ms

Round trip time grows to 100ms, splitting it in: 10ms for edge's paths and 30ms for the bottleneck. This scenario responds to the usual Internet usage, for which AQM algorithms establish their default values. Thus, is expected to have the best and the most linked to the theoretical values.

Like the previous section, no congestion scenario generates the same result for every queue management method. However, it is useful for know the base numbers at the given path and queue capacity. The table above, Table 4 , presents this values:

Table 4. Scenario 2 Results Values

Algorithm	Avg. Throughput	Avg. Goodput	Mean Delay	Mean Jitter	Drop Packets
All	3.20 Mbps	2.53 Mbps	52.94 ms	1.14 ms	0

Comparing to the previous situations, logical changes have been released. First, For the same injection throughput, less goodput has been scored due to the distance of the path. In addition, for the same reason, the average delay has incremented, however the jitter maintains the same value. This occurs because jitter does not depend on the path distance, it measures the time variability between signals.

On the other hand, the throughput and round trip time scored by the scenario are the ones displayed in the Figures 26 and 27.

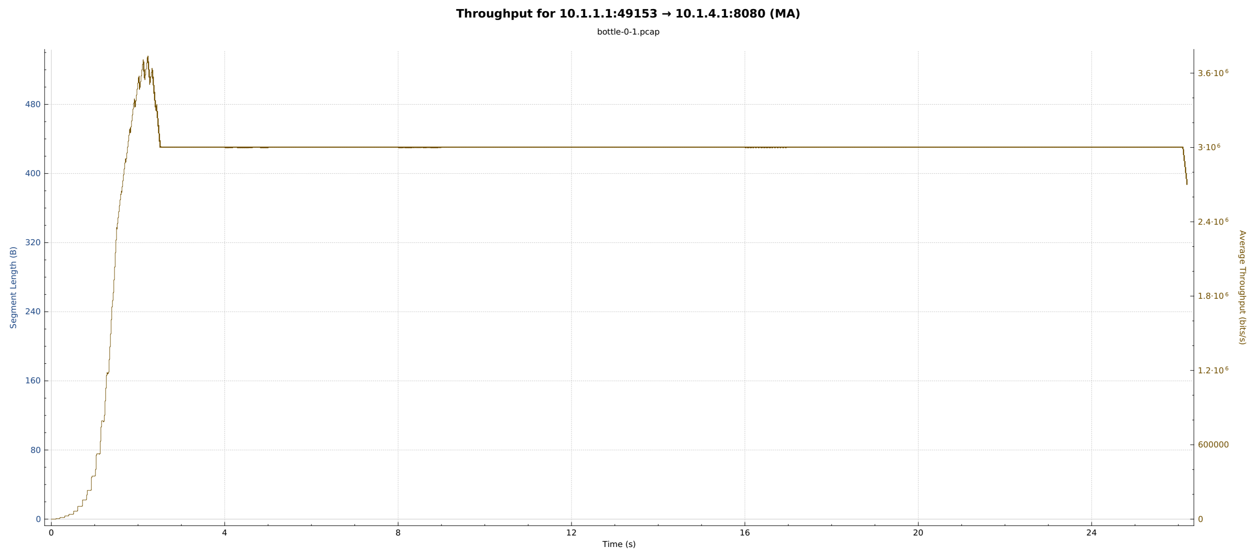


Figure 26. Throughput Drop Tail Scenario 2

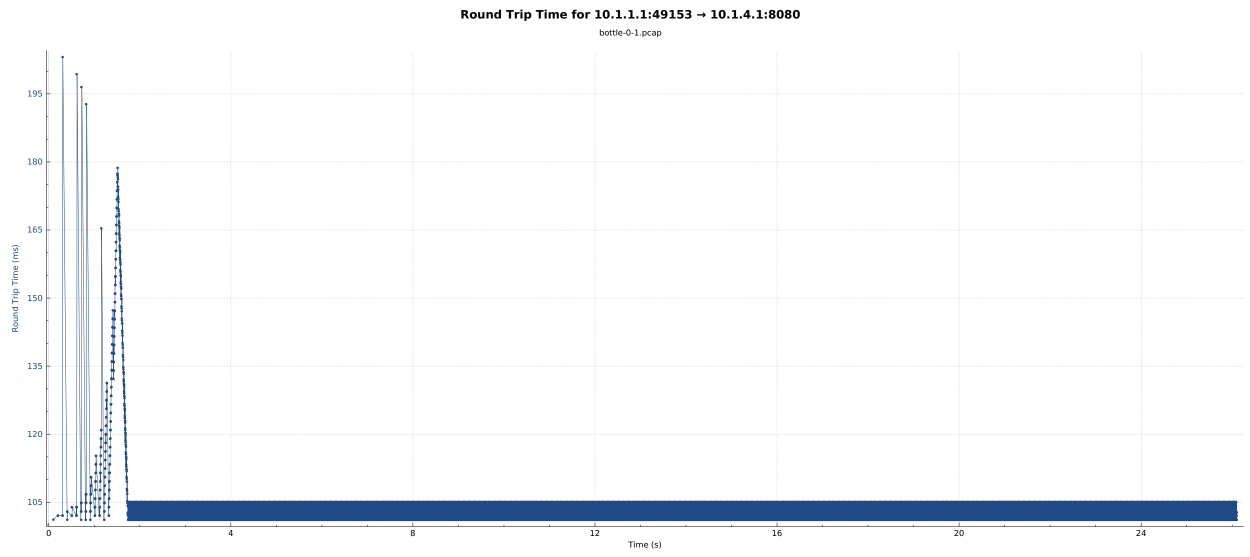


Figure 27. RTT Drop Tail Scenario 2

The values of both are the ones that Table 4 shows, showing the behavior of the traffic during the simulation time. Both graphics and table will be used as base for other scenarios.

Scenario 3: RTT = 400 ms

The 400ms of RTT are divided as follow: 50ms path for edges and 100ms path for the bottleneck link. Just as previous sections, Table 5 is the result's brief of the simulations. All the algorithms have the same response due to the congestion no existence, thus the values of the table are useful to take into account for further evaluations.

Table 5. Scenario 3 Results Values

Algorithm	Avg. Throughput	Avg. Goodput	Mean Delay	Mean Jitter	Drop Packets
All	2.29 Mbps	1.83 Mbps	202.42 ms	1.14 ms	0

Note the throughput lowering in contrast to scenarios 1 and 2. Increasing the path longitude, therefore, generates, obviously, a throughput decrease. Also, delay value has increased proportionally, maintaining, as explained before, the jitter with the same values. Hereafter, Figure 27 shows the RTT feedback of the simulation:

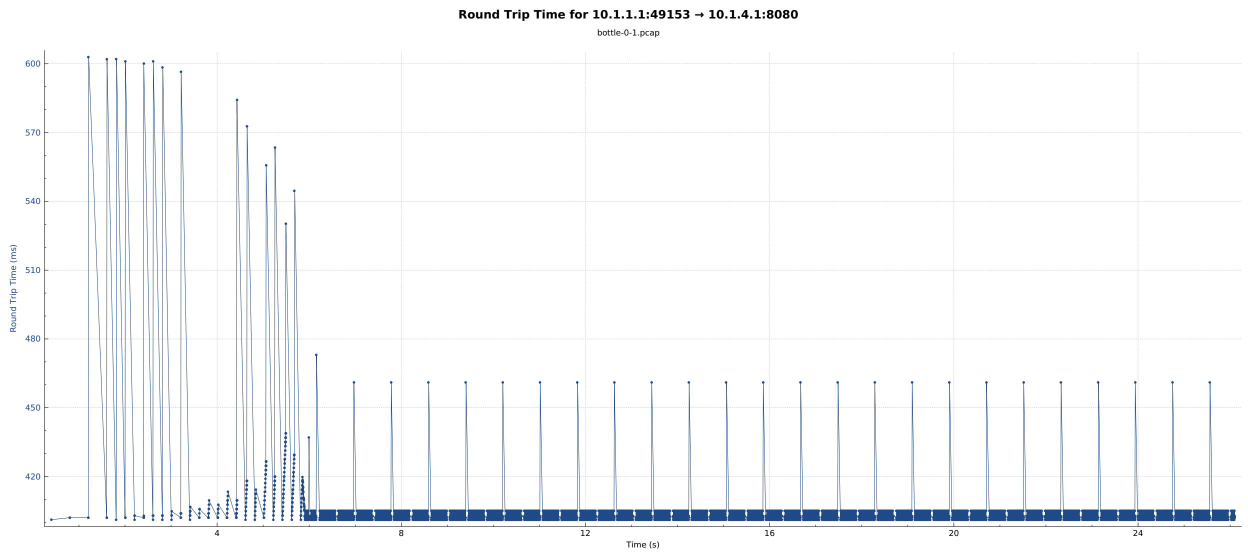


Figure 28. RTT Drop Tail Scenario 3

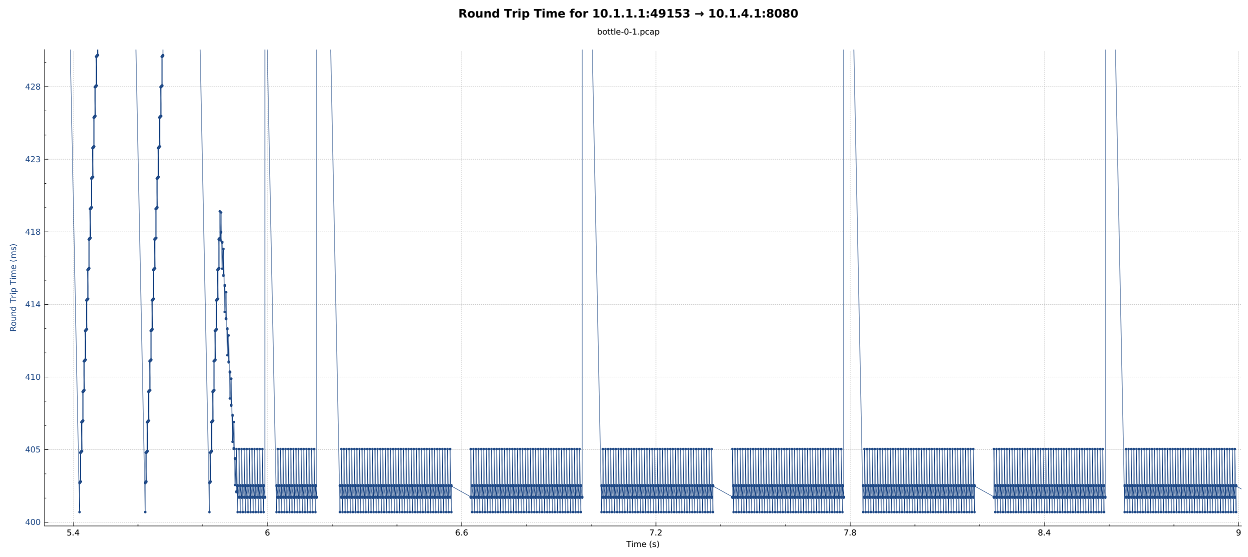


Figure 29. RTT Drop Tail Scenario 3 (zoomed)

At this situation, avoiding spikes pattern caused by the distance of the RTT (because it tries to maintain sojourn time below the target and starts dropping process, asking for retransmissions and decreasing the link utilization.), the latency values correspond to the total delay. Now, the stability takes longer time, around 6 seconds (comparing to previous sections of 2 seconds), therefore large paths span the convergence time. For easier visualization of the problem, Figure 30 shows how slow start time takes longer, about 6 seconds. Additionally, throughput behavior in TCP flow is more unstable.

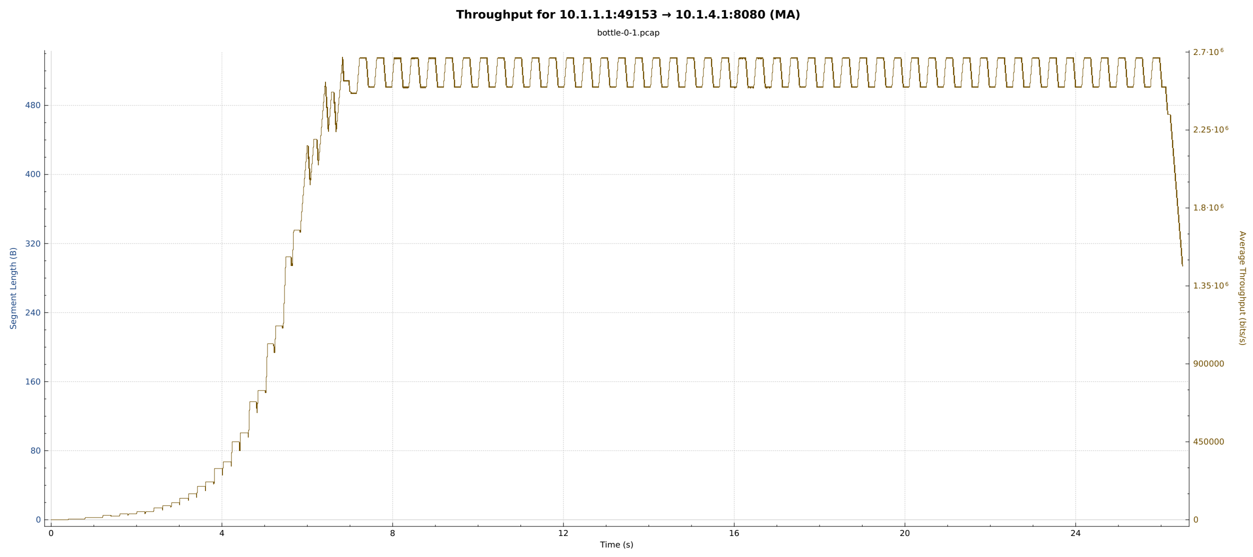


Figure 30. Throughput Drop Tail Scenario 3

Therefore, large round trip time simulations will give another point of view for the evaluation of AQM algorithms. In fact, this intercontinental large time values could approach to real situations, being useful feedback the AQM models behavior.

6.5.2 TCP congestion

Scenario 4: RTT = 50 ms

Now medium congestion will be set in the path. For it, TCP large flow is generated with 6 Mbps throughput, from node n0 to node n2. This scenario will evaluate the algorithms against elephant flow, with no variability of the traffic, only seeing the behavior against the bufferbloat problem. In fact, bufferbloat will be simulated in the drop tail case, checking the packets in a queue and the sojourn time of packets.

Table 7 shows a summary of the algorithms. We should underline the mean delay of each algorithm, being clearly superior to the drop tail one. In addition, dropping behavior could be checked: drop tail drops all the packets before the queue, in contrast, CoDel and FQ_CoDel drops some algorithms after the queue. Overall, FQ_CoDel and CoDel act in the same way, this is because FQ_CoDel with one flow acts identical (except one point) as CoDel.

Table 6. Scenario 4 Results Values

Alg.	Avg.Throughput	Avg. Goodput	Mean Delay	Drop p. before q.	Drop p. after q.
Drop Tail	4.88 Mbps	3.89 Mbps	123.46 ms	63	0
CoDel	4.88 Mbps	3.87 Mbps	86.8 ms	54	5
FQ_CoDel	4.88 Mbps	3.88 Mbps	93.3 ms	0	5

Bufferbloat problem is clearly shown in this simulation. Bellow, Figure 31 and 32 show the behavior of TCP in queues. After the slow start, the queue starts filling, until it gets full. In this case, after 15 seconds, sojourn time and packets in the queue starts increasing, until (in this case), TCP traffic stops. TCP sends data at 4.8 Mbps, 600 KBytes at one second, and the receiver (n4) could work without queuing until 15 seconds of senders deliver, after that, the queue starts filling as explained before. At last, TCP application stops at 27 seconds, not being the queue fully filled, but it is clearly displayed bufferbloat problem.

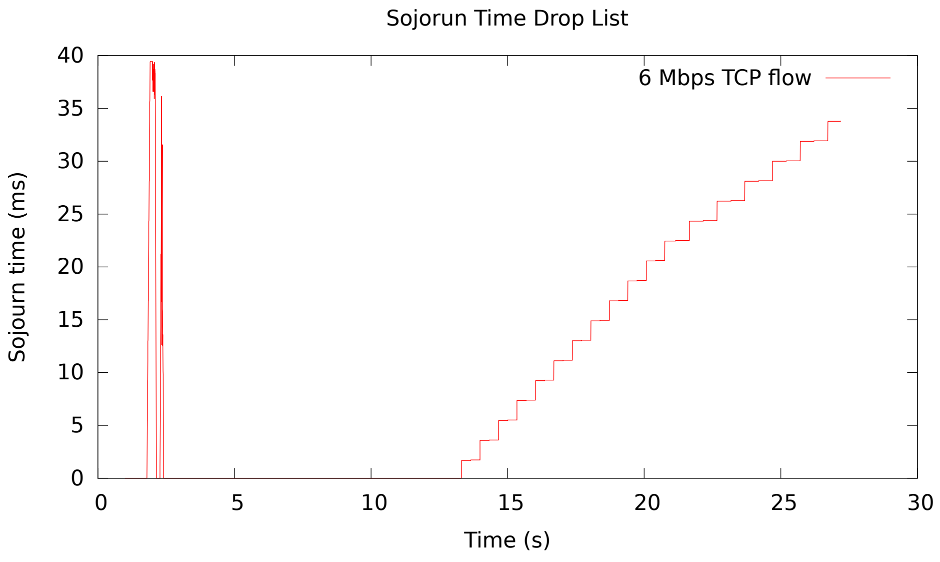


Figure 31. Sojourn Time Drop Tail Scenario 4

Figure 31 shows the increase of the sojourn time for packets at the queue, they gradually increase the time while the congestion increases at the path. Later the total round trip time, queue delay plus all other delays will be showed.

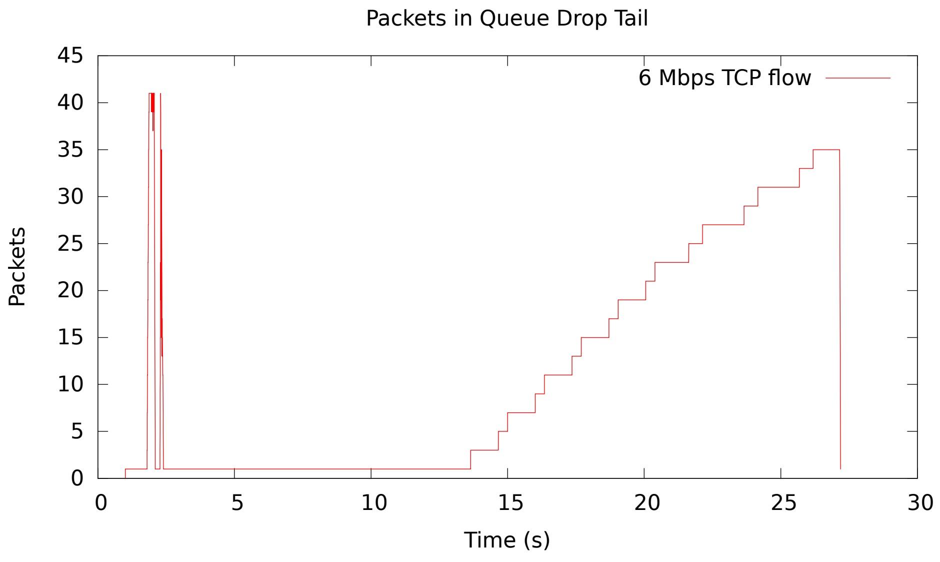


Figure 32. Packets in Queue Drop Tail Scenario 4

Figure 32 shows the packets increase at n4 queue. Highlight the maximum of 42 packets capacity in the queue. This is due to establish $2 * BDP$ size of buffer in the node queue. After the second 15, the queue is not fully filled, but it is clear the tend of getting filled, in other words, it is seen the bufferbloat problem.

Drop Tail, maximizes throughput at delay's expense. Figure 33 graphs the throughput of the flow, with a stable value of 4.58 Mbps after the slow start process.

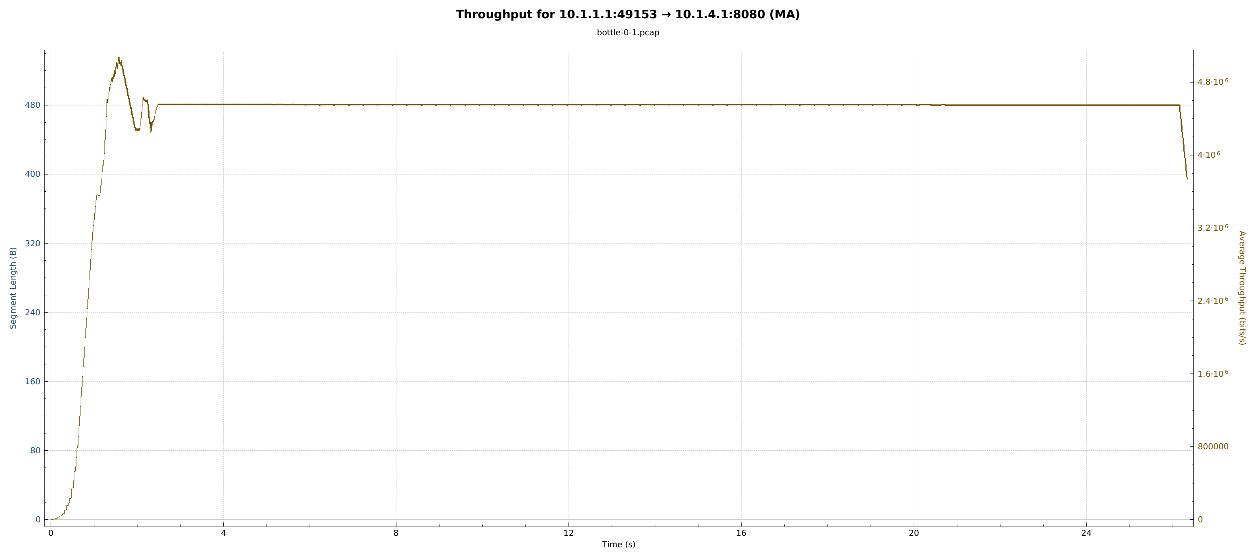


Figure 33. Throughput Drop Tail Scenario 4

On the other hand, round trip time increases during the simulation, as it is shown in Figure 34, being above 120 ms after 4 seconds of traffic. It displays how heavy the escalation of delay is, affecting directly to the packets of all traffic and, therefore, to the user's experience.

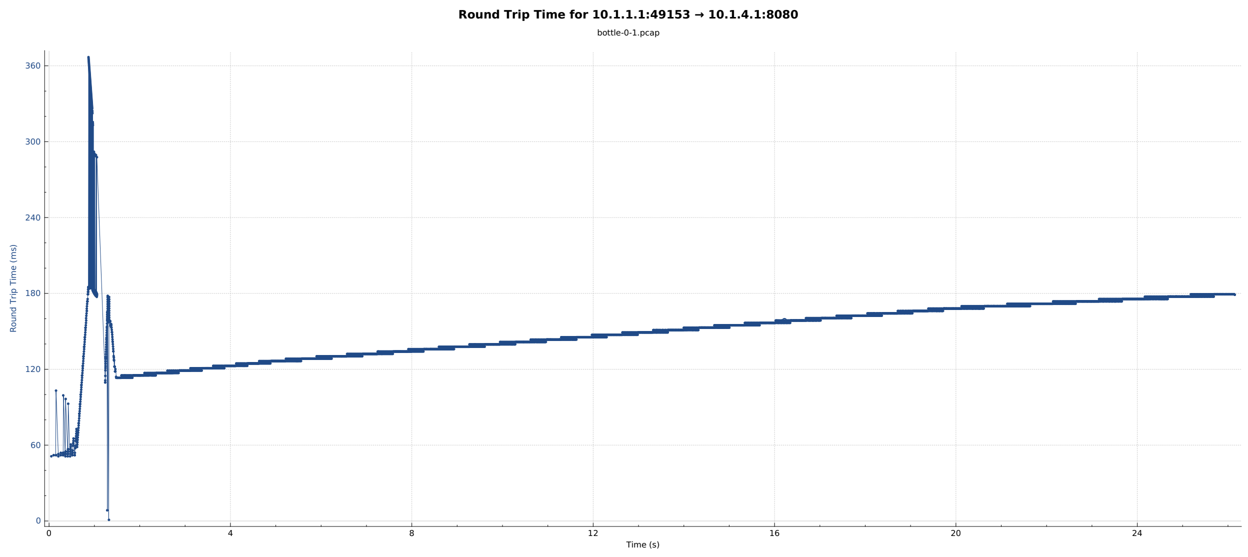


Figure 34. RTT Drop Tail Scenario 4

After explaining and checking bufferbloat problem, is time to evaluate the behaviors of the algorithms for the congestion situation. For that purpose, sojourn time and packets in queue comparison between Drop Tail and Codel will be an excellent point of reference.

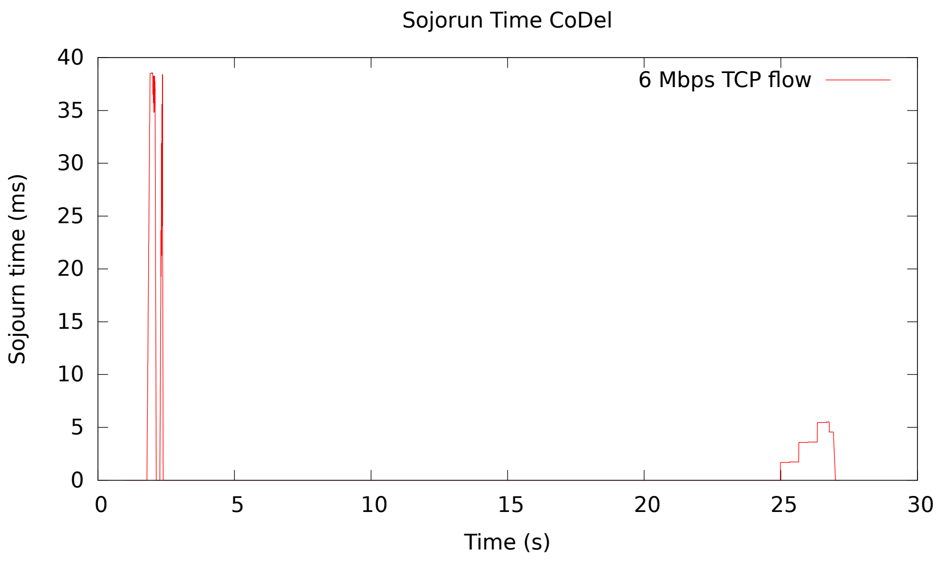


Figure 35. Sojourn Time CoDel Scenario 4

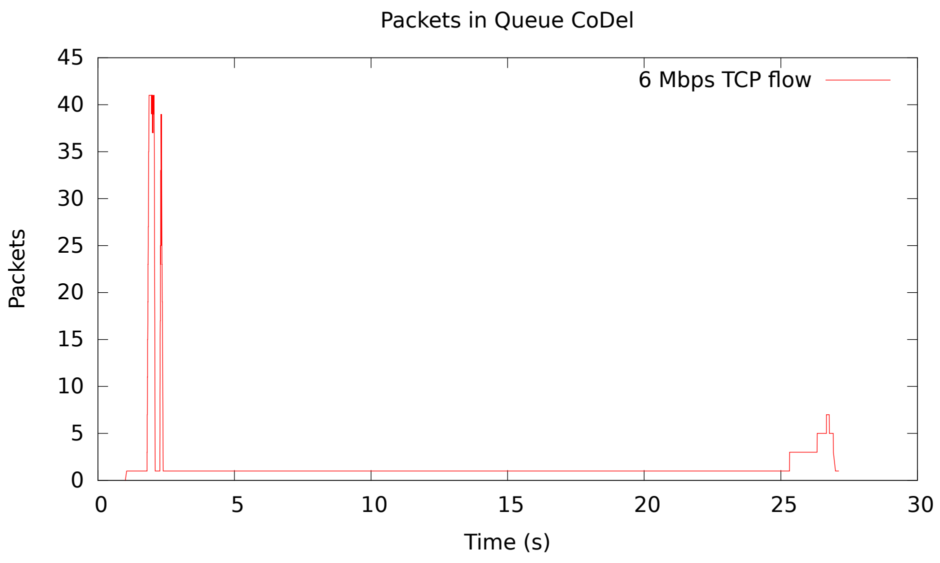


Figure 36. Packets in Queue CoDel Scenario 4

Avoiding slow start, CoDel results are strongly better than Drop Tail. Remember that CoDel, by default, has the TARGET value established at 5 ms of sojourn, as it showed in Figure 35 around the second 26. On the other hand, Figure 36 shows how the bufferbloat problem is a fight with AQM algorithm, maintaining the queue almost empty.

Finally, previously commented gap between CoDel and FQ_CoDel is presented. The graphic of Figure 37 compare CoDel and FQ_CoDel queue filling by bytes. After slow start, CoDel starts again with the bytes increase, while FQ_CoDel does not go to the start and uses the packets queue that has inside the buffer. Apart from this, they do not have differences of behavior in one flow scenario.

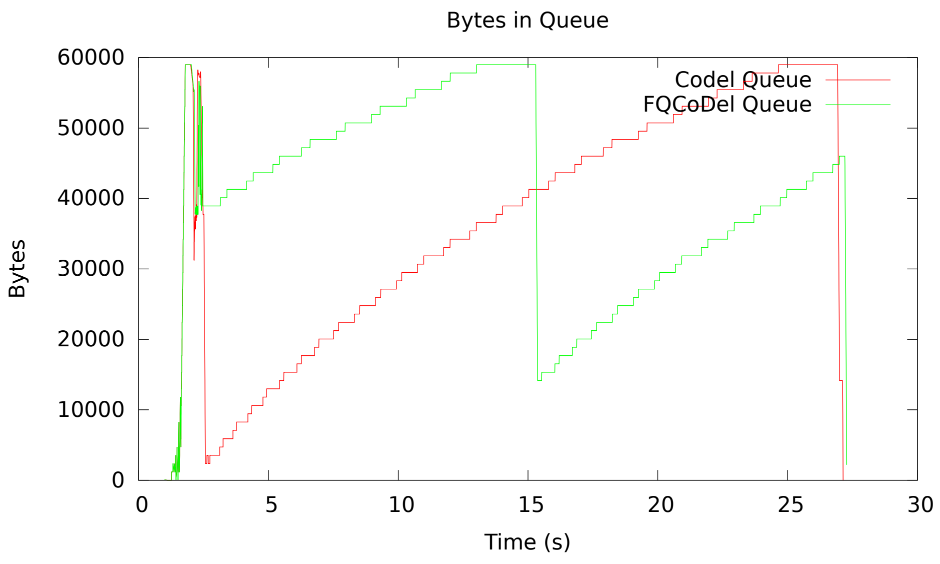


Figure 37. Bytes in Queue CoDel and FQ_CoDel Comparison

Scenario 5: RTT = 100 ms

According to the results of the previous sections, this one will be related to the characteristics of them but increasing the difference between CoDel and Drop Tail. FQ_CoDel, equal to the last section, works as same as CoDel when only one flow of traffic is generated, thus, showed results will summarize FQ_CoDel and CoDel in the same graphic.

Table 7. Scenario 5 Results Values

Alg.	Avg.Throughput	Avg. Goodput	Delay	Drop p. before q.	Drop p. after q.
Drop Tail	4.76 Mbps	3.78 Mbps	177.46 ms	63	0
CoDel	4.76 Mbps	3.78 Mbps	98.8 ms	0	4
FQ_CoDel	4.76 Mbps	3.78 Mbps	98.8 ms	0	4

The main difference between Drop Tail and CoDel is the latency values. With low congestion, like this case, the delay increases notably at Drop Tail model, while CoDel (equal as FQ_CoDel) maintains excellent results. Figure 38 present how Drop Tail latency increases heavily when the bufferbloat problem happens. On the other hand, CoDel (and FQ_CoDel) maintains the timing dropping packets when it goes above the TARGET sojourn.

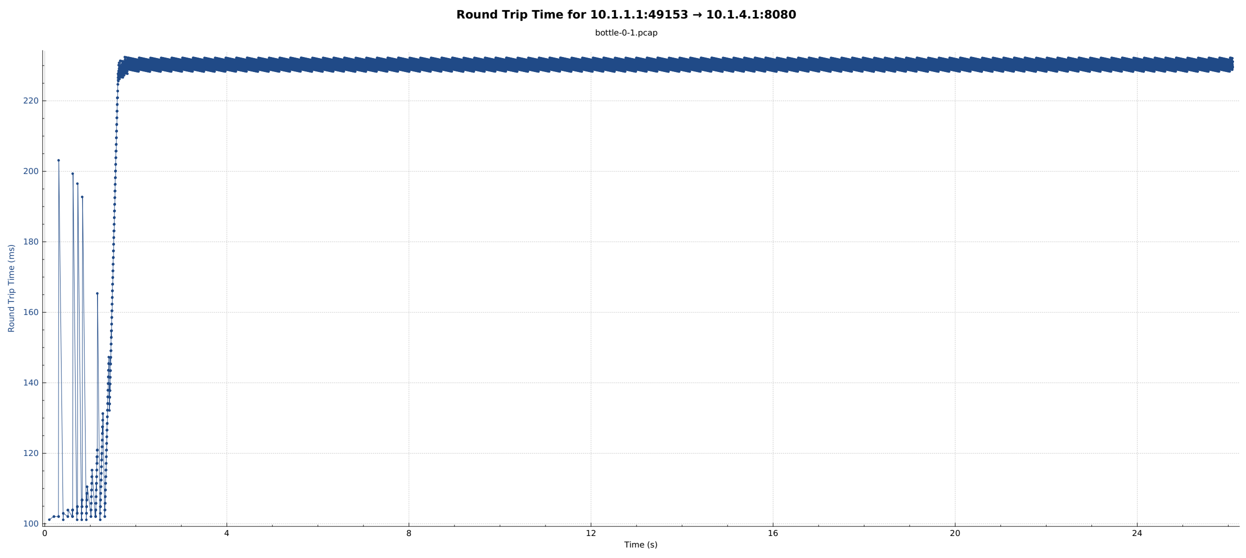


Figure 38. RTT Drop Tail Scenario 5

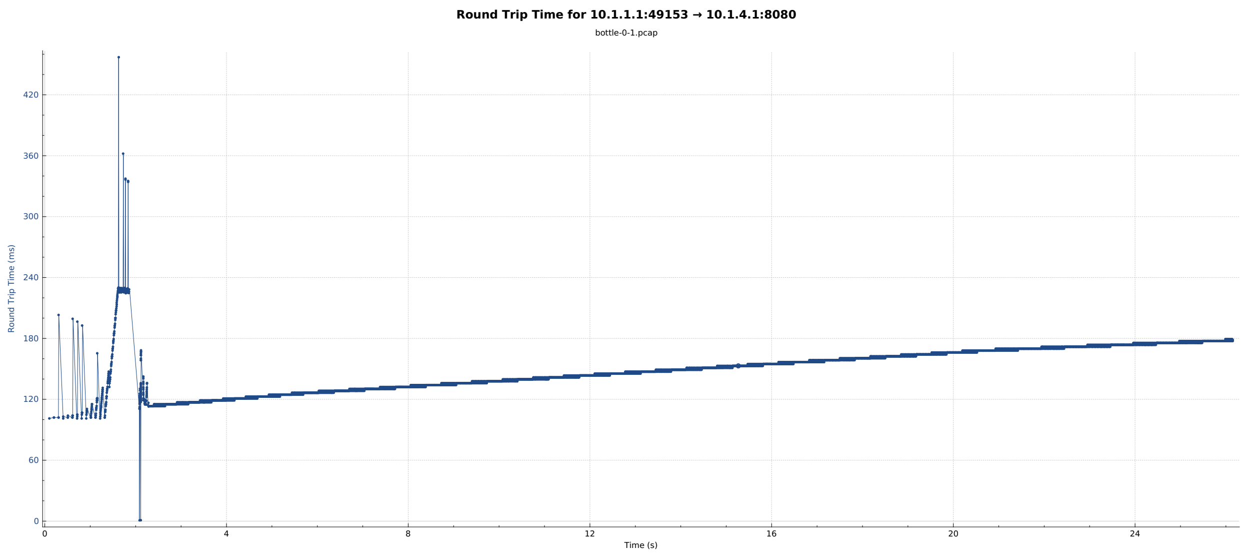


Figure 39. RTT CoDel and FQ_CoDel Scenario 5

In detail, sojourn time is compared in Figure 40. It is clearly defined how Drop Tail case fulls all the buffer and maintains the sojourn to the maximum always, while CoDel increases the sojourn time for milliseconds and the maintains the usual value, providing really good feedback.

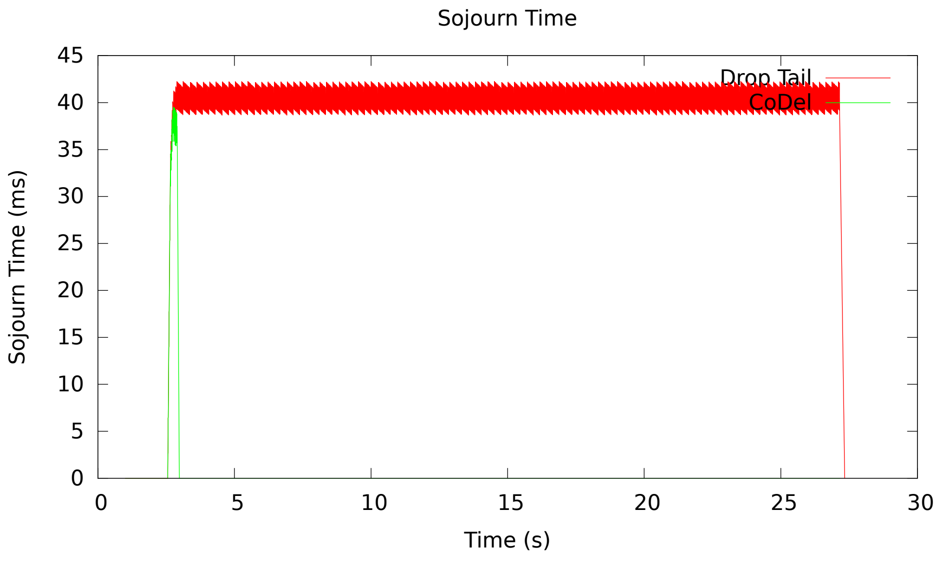


Figure 40. Sojourn Time Comparison Drop Tail and (FQ)CoDel at Scenario 5

Lastly, packets bytes in the queues are compared in Figure 41. As explained previously, CoDel and FQ_CoDel have the same result, nevertheless, Drop Tail result is strongly notable at the graph. Again, bufferbloat problem is showed, since Drop Tail maintains the queue full all the simulation time.

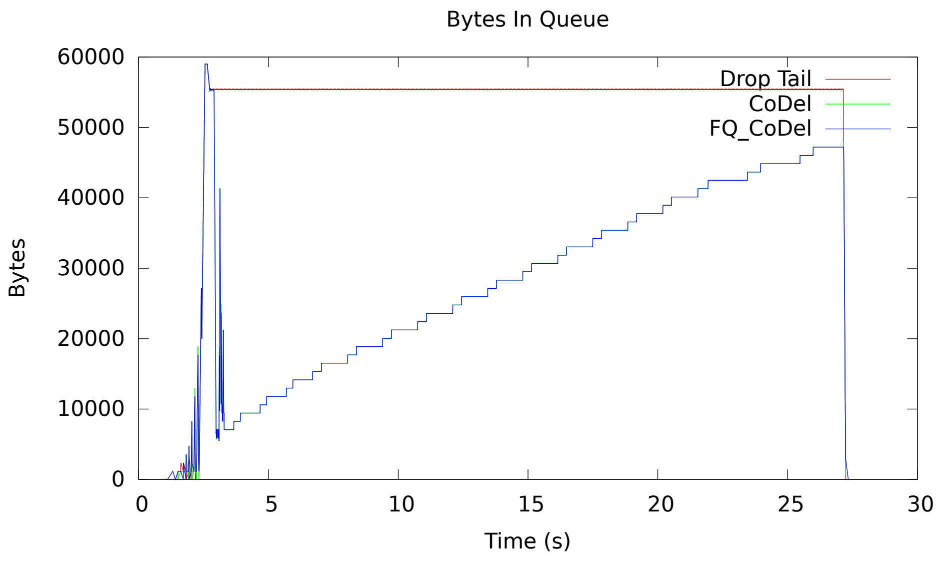


Figure 41. Bytes in Queue Comparison in Scenario 5

This scenario is the one that better samples the theoretical points. The round trip time value is the ideal one for the algorithms, and the low congestion permits see the difference between CoDel, Drop Tail and bufferbloat problem.

Scenario 6: RTT = 400 ms

This scenario breaks with the previous sections in congestions situations. Now, 400 ms of RTT generates a larger queue buffer hardware, taking account of the rule of thumb. That is to say, $2 * BDP$ generates 5 MBytes buffer in the bottleneck, much larger than previous ones against the same throughput of the TCP flow. In this case, as we could see in the next figures, there is not congestion, being not triggered buffer management mechanism. This occurs because 6 Mbps throughput is not enough to fill in 30 seconds the 5 MBytes buffer installed, according to the $2 * BDP$ rule, at the queue. In contrast, if the queue would be filled, queue delays would be enormous, as happened when manufacturers started installing larger buffers in the nodes thanks to the increase of the chips capacity (and price decrease) of Moor's Law. Table 8 shows the values obtained from the simulation. They are almost the same from Scenario 3.

Table 8. Scenario 6 Results Values

Algorithm	Avg. Throughput	Avg. Goodput	Mean Delay	Mean Jitter	Drop Packets
All	2.29 Mbps	1.84 Mbps	203.04 ms	1.14 ms	0

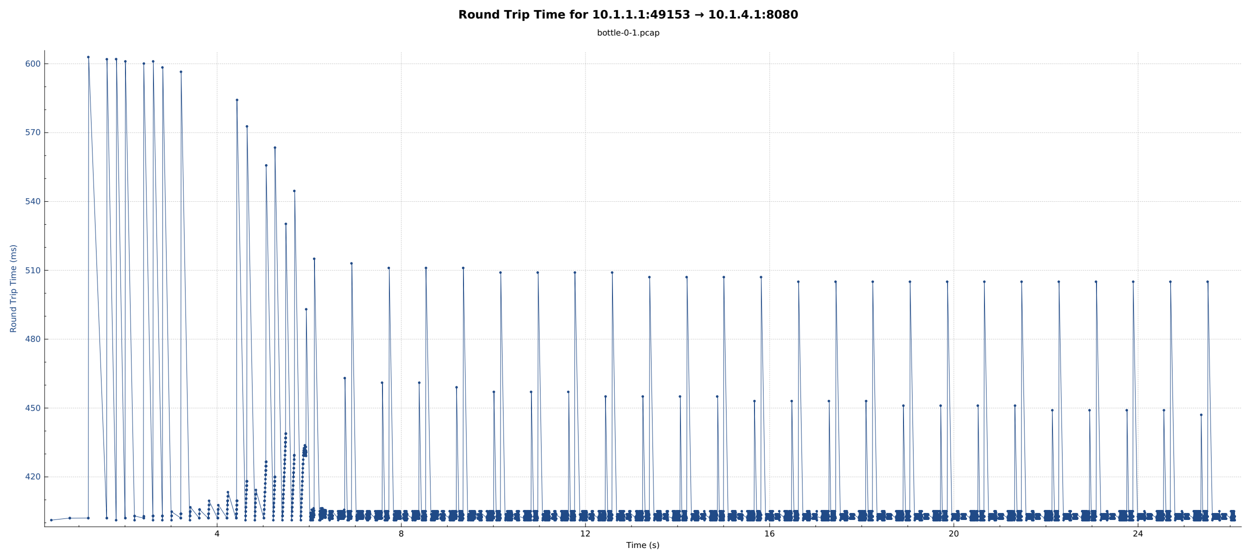


Figure 42. RTT of the Scenario 6

At last, simulations shows how (Figure 43), not being congestion all the installed AQM models acts in the same way, with the same queue bytes buffer fill.

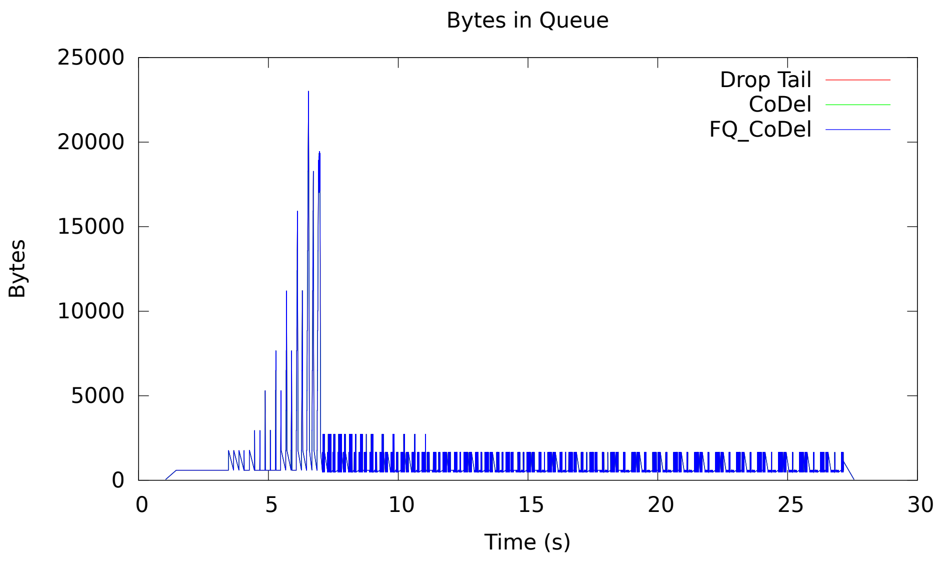


Figure 43. Bytes in Queue Comparison in Scenario 6

6.5.3 TCP + UDP congestion

Scenario 7: RTT = 50 ms

At this point, mixing TCP and UPD flows will generate a heavy congestion situation. In detail, it consists of TCP elephant flow of 4 Mbps throughput, mixed with 4 Mbps throughput of UDP mice flow. TCP flow sends 10 MBytes of data, while UDP only 5 KBytes, the conversation will be established from n0-n2 for TCP and n1-n3 for UDP, both through the 5 Mbps bottleneck. UDP flow will be created from the 10 to 20 seconds, but TCP flow will be running from 0 to 27, like in other cases.

Like previous situations, 50 ms round trip time with 42 packets for queue capacity is installed at n4 queue, being the RTT lower than default CoDel's INTERVAL value. At this time, FQ_CoDel has a different behavior from CoDel, it split flows in queues, applying flow isolation and priority to new flows, applying CoDel algorithm for each one. This, as will be seen below, will offer better results and higher link utilization.

The evaluation of both of the main algorithms will be done checking the latency and throughput of the TCP flow. It will be harmed during UDP flow traffic, from 10 to 20 seconds, as it is seen in Figure 44 and Figure 45.

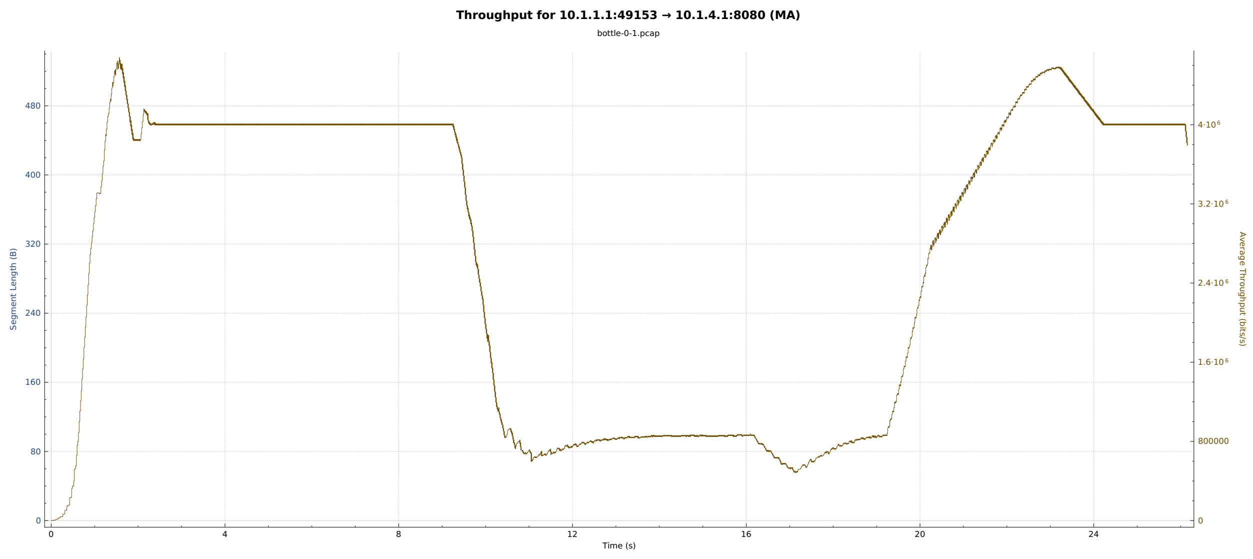


Figure 44. Throughput CoDel Scenario 7

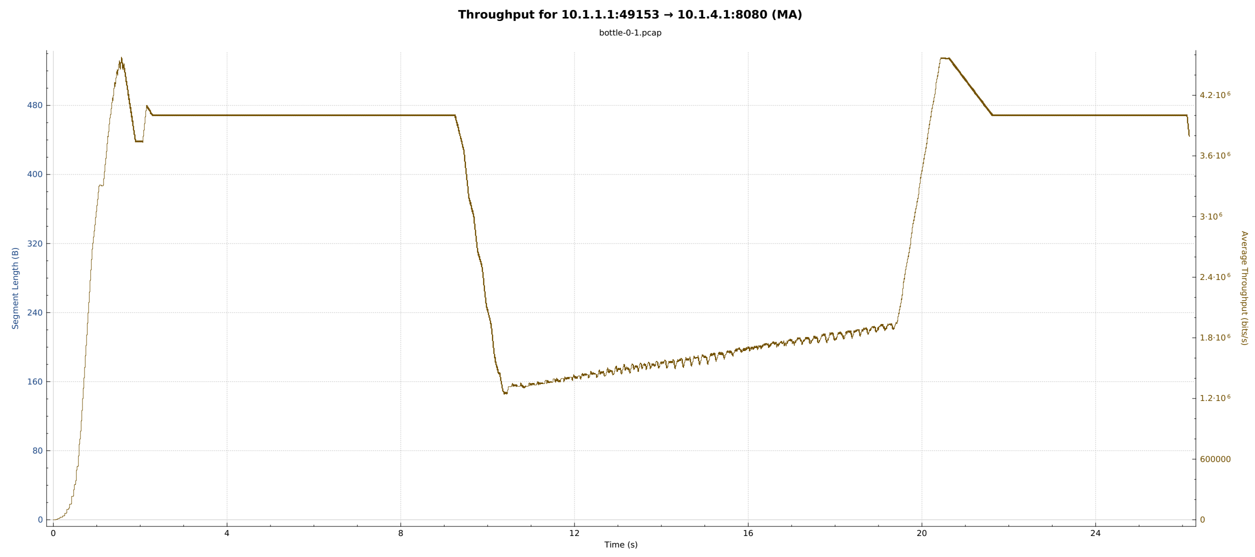


Figure 45. Throughput FQ_CoDel Scenario 7

The Figures show the variation of throughput during traffic mixing time. CoDel decreases notably the value, reaching values under 1 Mbps. On the other hand, FQ_CoDel decreases the value to 1.4 Mbps but it starts increasing it smoothly until the UDP flow finish. It provides a better result, establishes values and better than CoDel by itself. This is due to the flow isolation provided by FQ_CoDel, however, CoDel's behavior is better with larger RTT, as it is seen bellow.

In consideration of latency, FQ_CoDel provides better feedback. Avoiding spikes, the latency of CoDel works around 200 ms and in not so establish mode, while FQ_Codel provides lower latency values with almost all the time being the round trip time of 180 ms.

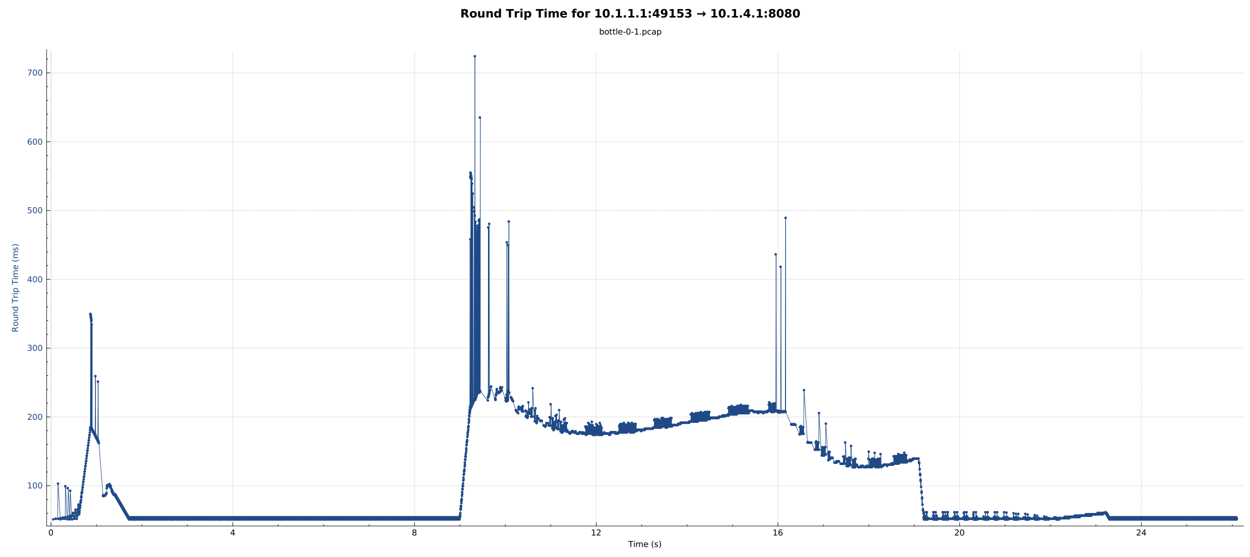


Figure 46. RTT CoDel Scenario 7

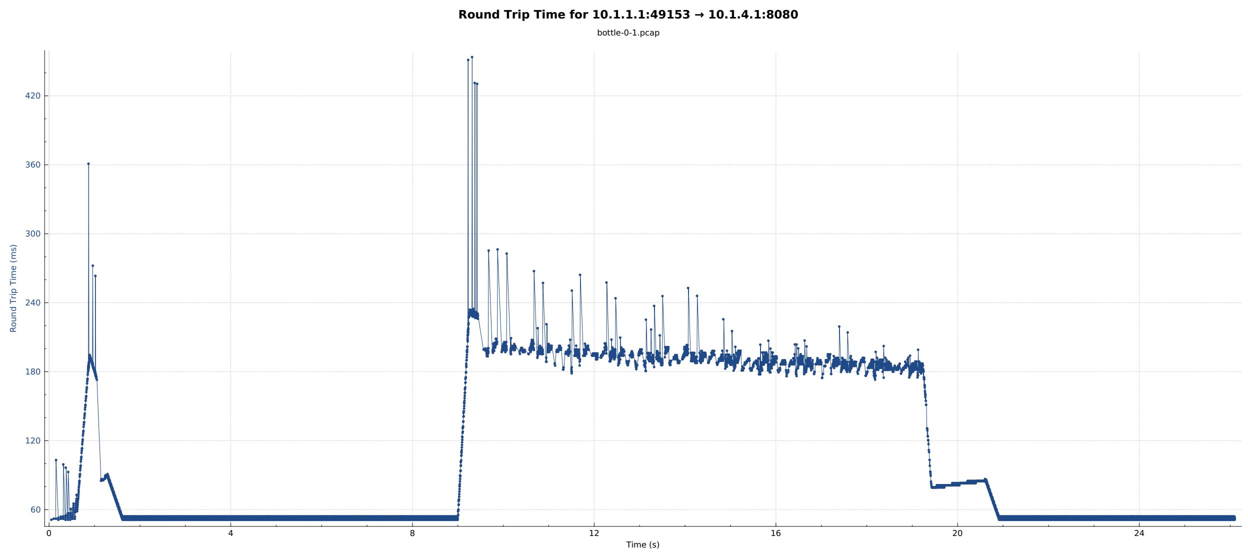


Figure 47. RTT FQ_CoDel Scenario 7

FQ_CoDel, therefore, offers better results to latency in mixing flows situations, isolating each flow and offering CoDel results to each one independently. However, is necessary to check the behavior of it in larger round trip time situations.

Figure 48 shows the evolution of the bottleneck queue capacity during the simulation. Drop Tail has the worst adaptability to the traffic mixing, being the one that most time is at the maximum of the queue. CoDel, for his part, offers better quality, but not being too much reliable. Finally, FQ_CoDel result‘s are below the queue limit, with more stable evolution of the queue, in other words, with less drastic packet dropping.

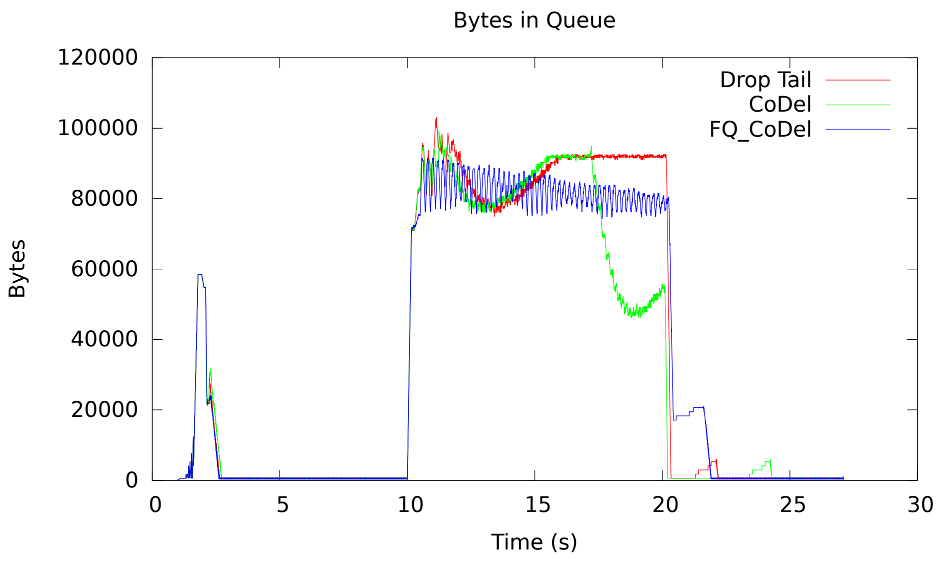


Figure 48. Bytes in Queue Scenario 7

Scenario 8: RTT = 100 ms

Scenario 8 works with mixed traffic in usual Internet connection round trip times. Table 9 summarizes the results of the simulations done. Drop Tail displays the worst latency values, being followed by Codel algorithm. However, Drop Tail's throughput scores the greatest values due to TCP Reno which maximizes always the throughput, harming other transport protocols such as UDP.

Table 9. Scenario 8 Results Values

Alg.	Avg.Throughput	Avg. Goodput	Delay	Drop p. before q.	Drop p. after q.
Drop Tail	2.79 Mbps	2.22 Mbps	82.68 ms	208	0
CoDel	2.51 Mbps	2.00 Mbps	71.05 ms	0	24
FQ_CoDel	2.53 Mbps	2.02 Mbps	62.22 ms	0	234

FQ_CoDel scores the best latency values on account of dropping packets after the queue, precisely, it drops 234 packets while CoDel only does it with 24 packets. This conduct helps mice flows to cross the bottleneck without suffering high delays. Furthermore, flow isolation of FQ_CoDel provides good throughput values, getting higher results than CoDel.

Round Trip Times of the flows is really a good point of comparison for the algorithms, thus we will use Wireshark .pcap graphics for analyzing the simulation of each one. The measures are taken from the TCP stream, using the round trip time measured in the queue ingress link. Figures bellow (49, 50 and 51) present the values during time simulation. First, notice that Drop Tail behavior in mixed flows is really poor, scoring pikes of more than 1.8 seconds of delay. Secondly, Drop Tail, after detecting the congestion starts dropping packets (before queue 208), generating another slow start and, in case of more than one TCP connection, global synchronization issue. On the other hand, gaps between CoDel and FQ_Codel are not totally significant. It is true that, after some period of time, CoDel acts more stable than FQ_CoDel, however, the delay spikes of CoDel are higher than FQ_CoDel. Finally, FQ_CoDel adapts itself to traffic changes better than CoDel, being the algorithm that better results offer at 100 ms RTT. Those indicators should take into account, mostly when more than two flows are working at the same time.

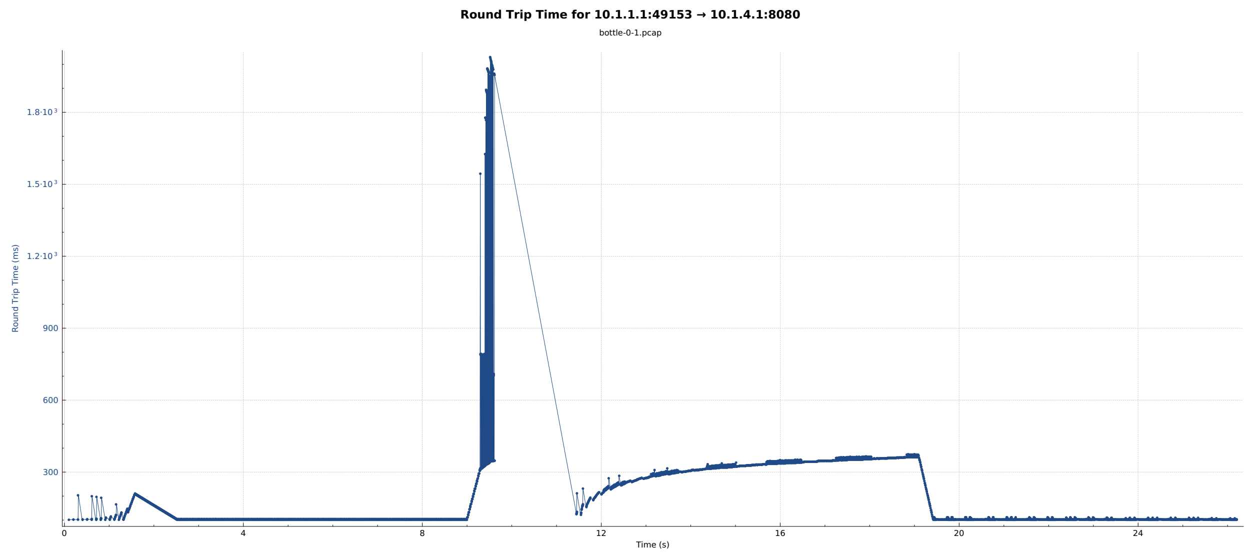


Figure 49. RTT Drop Tail Scenario 8

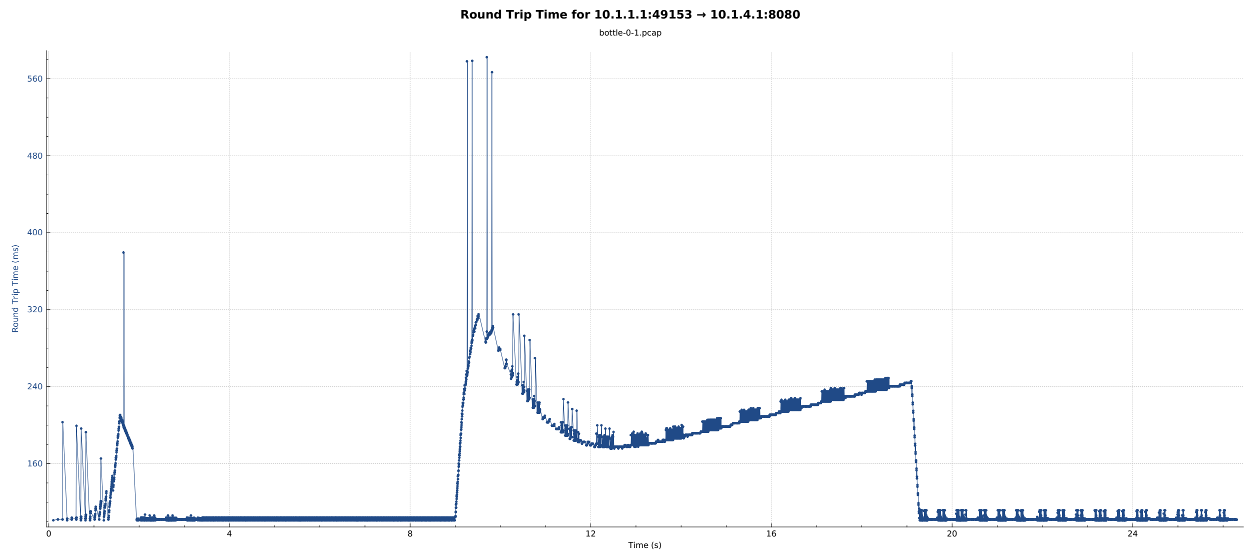


Figure 50. RTT CoDel Scenario 8

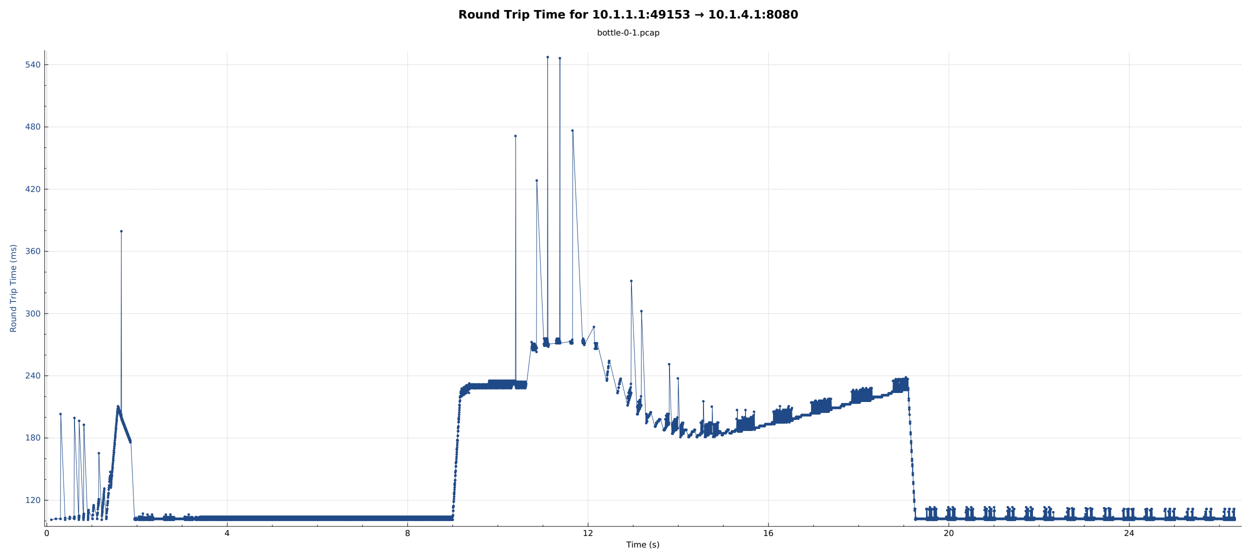


Figure 51. RTT FQ_CoDel Scenario 8

Lastly, as previous sections, Figure 52 compare the queue situation of the algorithms during the simulation. Feedback of bytes are really useful for comparing the AQM models, in fact, this time shows clearly the performance of each algorithm. Taking into account that Scenario 8 works at Internet RTT values and the congestion case is usual (large TCP flow against mice UDP flow), the graphic shows a common Internet's users situation.

First, check Drop Tail conduct. After slow start, the queue is not filled (working with TCP 4 Mbps throughput against 5 Mbps bottleneck), despite this after UPD flow injection queue starts filling quickly (4 Mbps of UDP plus 4 Mbps of TCP flow), reaching the limit of queue after few milliseconds. After that, Drop Tail starts dropping all the packets, making TCP flow starts again (around second nº 12). Then, the queue fills quickly again, maintaining stable high congestion during both flows intervention.

CoDel, on his part, acts to the traffic change quicker than Drop Tail and FQ_CoDel, decreasing the buffer fill fast below target point. However, the performance of the flow throughput is lower at that period. FQ_CoDel, works worst with congestion increase for 1 or 2 seconds but then it adapts to the congestion maximizing the throughput and minimizing the latency of the path.

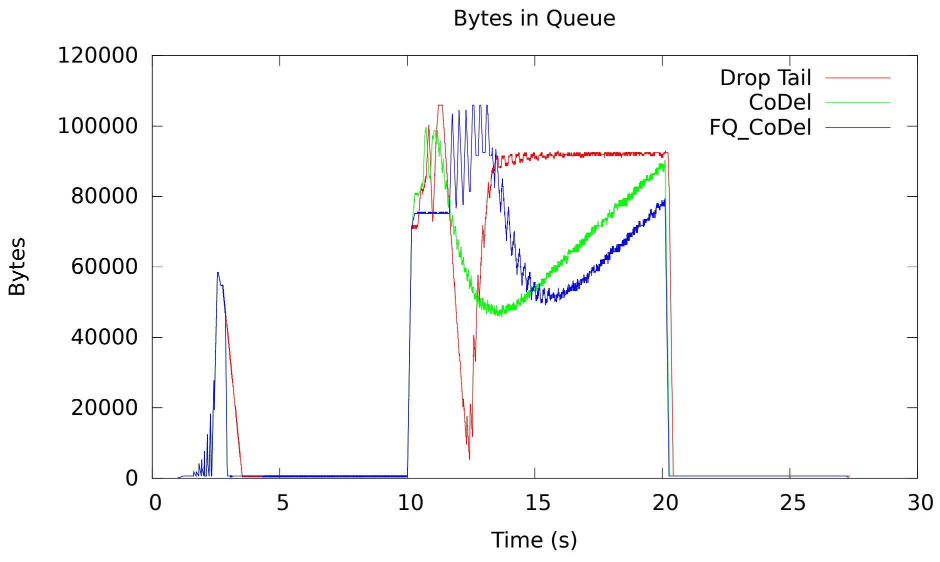


Figure 52. Bytes in Queue Comparison Scenario 8

Scenario 9: RTT = 400 ms

In the last Scenario, mix traffic will be simulated in a large round trip time path. Considering other RTT = 400 ms scenarios, queue buffer is large enough to accept all the throughput sent by n0. However, in this situation, congestion will come out. Since Drop Tail results are the worst and it is not worth it to compare them, we focus the results in CoDel and FQ_CoDel.

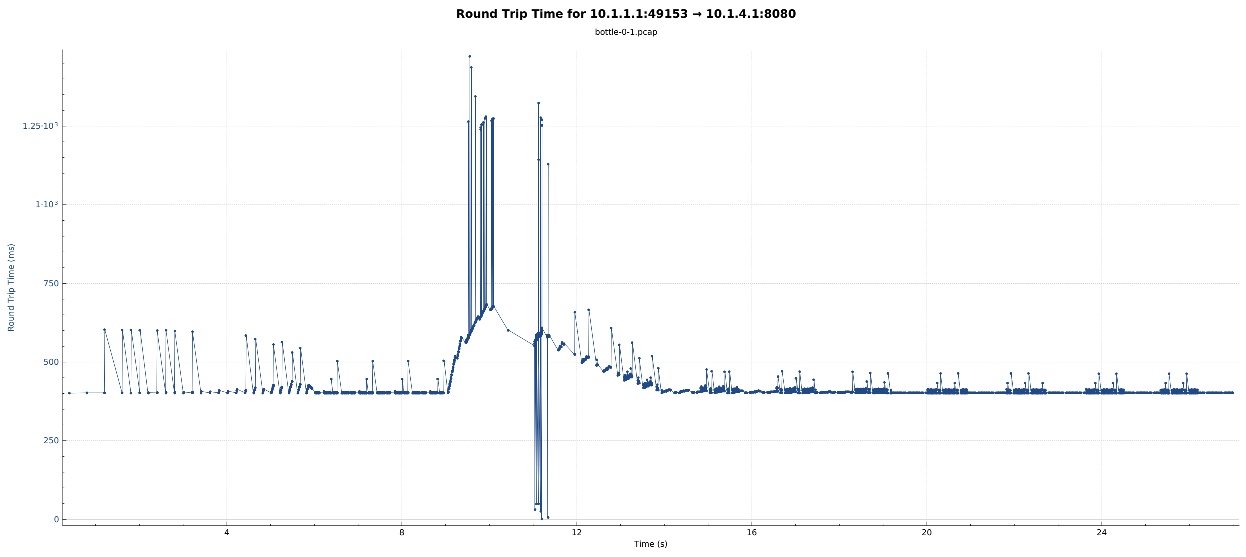


Figure 53. RTT CoDel Scenario 9

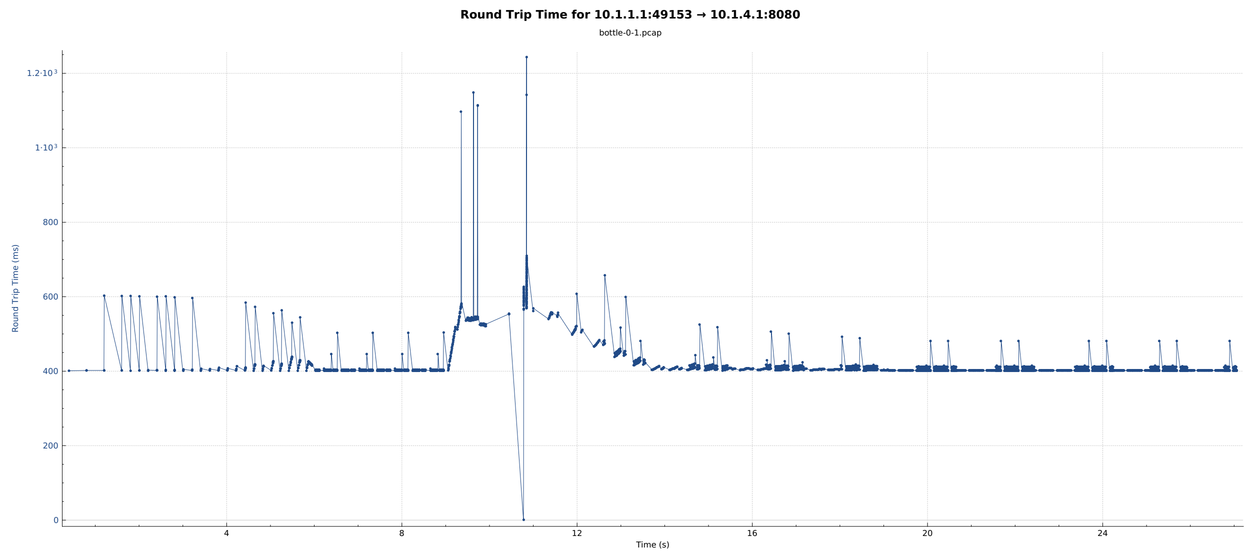


Figure 54. RTT FQ_CoDel Scenario 9

Latency values are similar in both cases, however, CoDel has more spikes and it reaches higher latency values. Just when mix flows overlaps (second nº 10), CoDel responds with 1.2 seconds of RTT, a really high value for delay. Even so, both AQM algorithms adapts quick to the flows, decreasing the latency to good values such as 400 ms, almost without delay in the bottleneck queue.

A detailed review of the behavior of algorithms could be done checking packets in queue graph. Figure 55 shows how the algorithms fill their queues during the simulation. Drop Tail, gets filled past 12 seconds of simulation, reaching the 333 packets of queue established by the $2 * BDP$, after that, it starts dropping packets trying to advise the sender about congestion at the link.

On the other hand, CoDel and FQ_CoDel fill the third part of the queue (100 packets) only for twos seconds, then, they manage the queue for achieving the expected results. They score similar results, hence checking the sojourn time of both queues could help to see the real time that packets spend inside the queue.

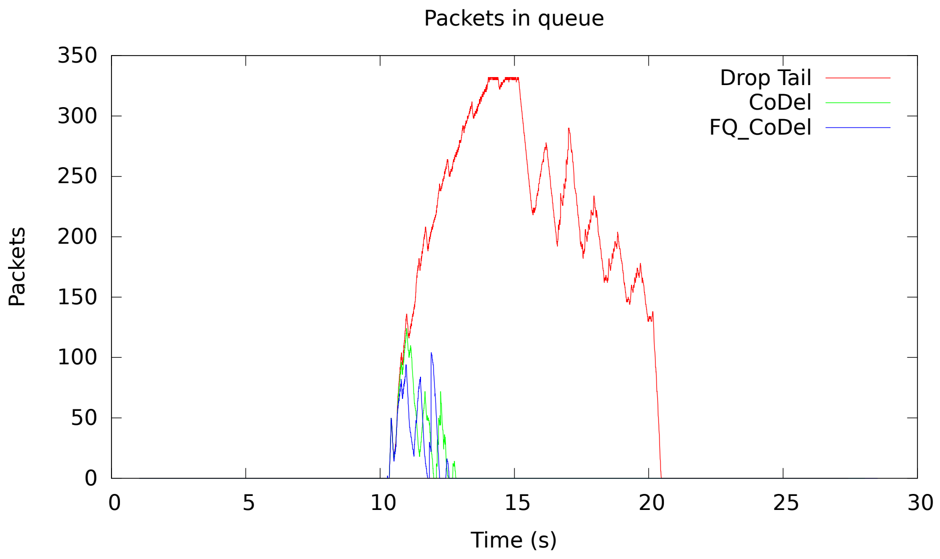


Figure 55. Packets in Queue Scenario 9

Figure 56 shows the time that packets have to spend inside the queue. FQ_CoDel has not really good response in sojourn time, providing worse results than CoDel. As explained in section 6.2, FQ_CoDel gives priority to new incoming flows, that is why the Figures 51 offers that values, due to the incoming UDP traffic. This could respond to bad utilization of the throughput and therefore the bad response in the sojourn time of the queue.

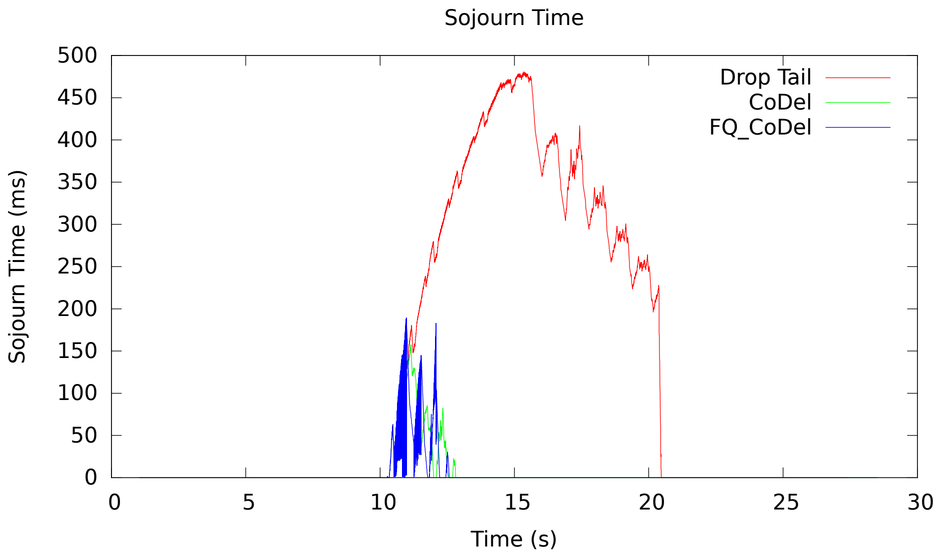


Figure 56. Sojourn Time Scenario 9

6.5.4 Overall Results

Combination of FQ_Codel and ABE is the proposal of the thesis, however, ns3 does not permit the simulation of these two algorithms mixture, making necessary the manual calculation of the expected results. In detail, [20] and [28] documents are used for the results considerations, using the percentages and values of the papers.

According to [20], the improvement of using β_{ecn} higher than the default 0.5 value, improves the throughput around 13% to 31% for TCP Reno flow with CoDel managed queue. Establishing 0.7 as useful β_{ecn} value, the exact percentage of improvement is 13% for 50 to 200 ms of RTT and 11% (with high variability) for higher RTT such, 400 ms. Appliying those values to our simulations, the scored result is showed in Figures 57, 58, 59.

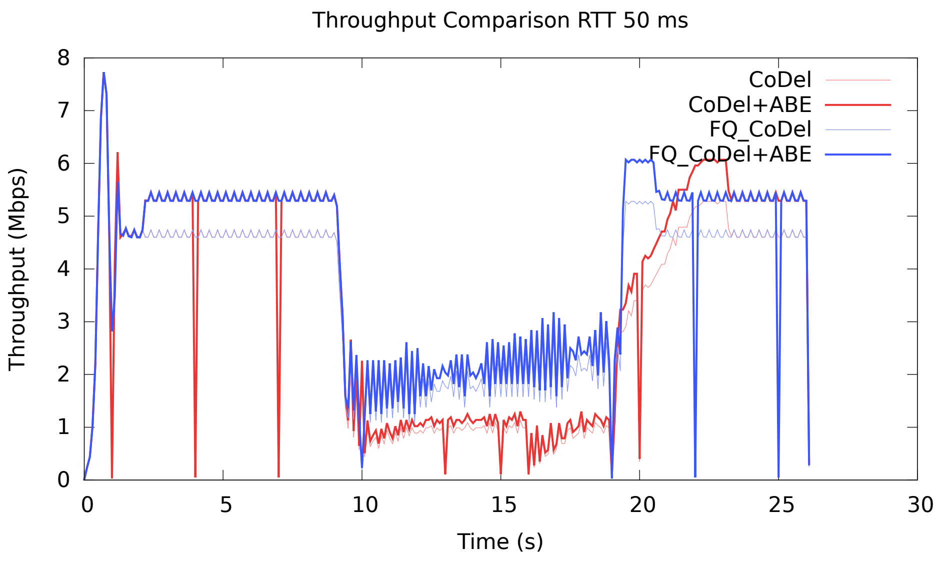


Figure 57. Throughput Improvement for RTT=50ms

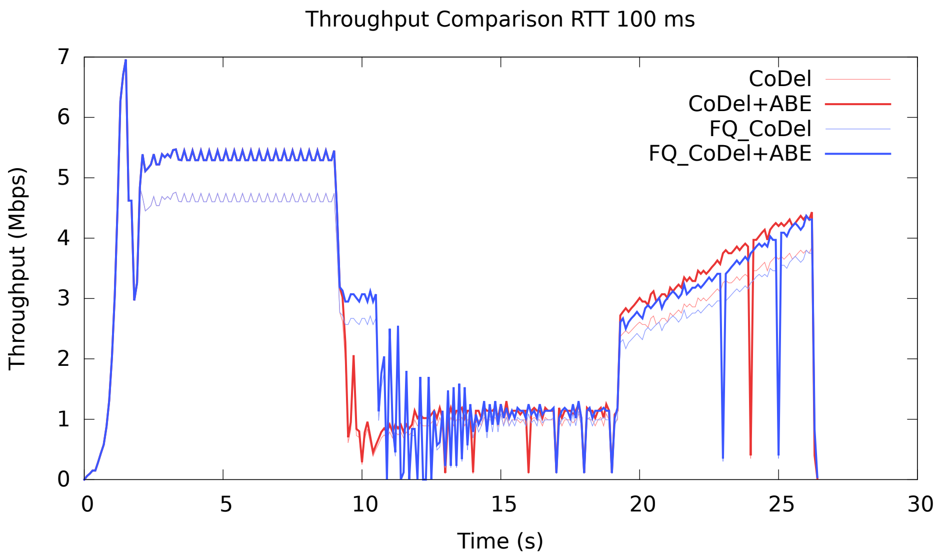


Figure 58. Throughput Improvement for RTT=100ms

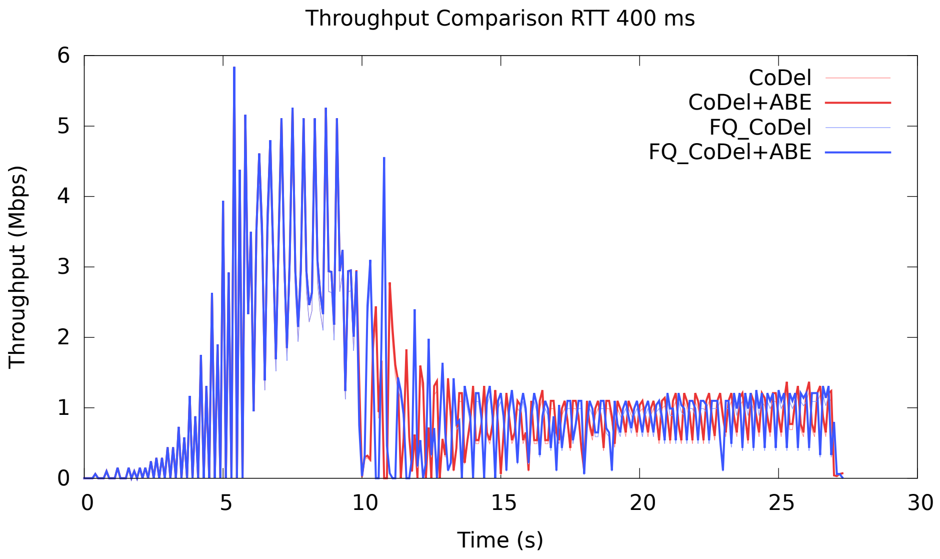


Figure 59. Throughput Improvement for RTT=400ms

Each case maintains the throughput progress of the simulation showed in previous sections. In general, throughput mean value for RTT=50ms increases from 2.96 to 3.34 for CoDel and 3.35 to 3.78 for FQ_CoDel; for RTT=100ms it changes 2.51 to 2.83 for CoDel and 2.53 to 2.85 for FQ_CoDel; finally for RTT=400ms, it increases from 0.99 to 1.09 for CoDel and FQ_CoDel. The throughput values are notable, making the utilization of the link higher.

Short flows are really harmed by the default $\beta = 0.5$ value, i.e. web traffic which may terminate shortly after the slow start, suffering the halving decrease. Additionally, is is a substantial increase in the use of short flows, adding an extra delay just for waiting for the after slow start congestion avoidance phase. ABE elude this effect and permits the high utilization of the link.

It is clearly shown in previous graphs how throughput increases for the AQM models, however, in return, it generates a 1ms of queueing delay for $\beta_{ecn} = 0.7$. Even so, the cost of delay worth it the increase of the link utilization.

Thesis proposal generates a valuable optimization of the AQM FQ_CoDel model use. Throughput improvement, about 11% to 13%, avoids the under utilization of the link and makes TCP's sawtooth smoother. Extra, it improves the flow finish time for short flows, whose terminate their send after the slow start. There are related works that uses this technique for low latency networks, such as, Data Center TCP (DCTCP) which also propose the update of TCP's ECN response. All the works show measurable benefits when networks change their ECN marking behavior, however, nowadays is ardour to make the change for all the senders and make active the ECN in all nodes on the path. The conclusions (section 8.1.3) of the work will detail all these points.

7 Methodology followed in the development of the work

7.1 Description of tasks, phases, teams or procedures

Table 10 shows the Work Breakdown Structure of the project. Hereunder, main phases are explained, indicating how they have been done and the principal points of each one.

7.1.1 Project Supervision

The first phase is the supervision of the project. It starts when the project starts and finish when it has been composed. In other words, it spans for all the project's duration, controlling that all phases are developed and managing deadlines dates.

7.1.2 Prior Preparation

First, the project is chosen between several alternatives. Once selected, the scope of the project is developed, deciding the points of study and filtering the information that is proposed to research.

7.1.3 Preliminary Analysis

Before researching the project's topic, it is crucial to get a knowledge of it, which will be closely linked to the AQM models comparative. Thus, is necessary to have a good base about queues and their issues prior to analyzing each algorithm and their characteristics.

7.1.4 Study Methods

This is the main phase of the project, where the evaluation itself is done. At this point, the different AQM techniques are researched, taking into account the main ones and those who other researches have highlighted recently. After that, the alternatives evaluation is done and, with it, the outstanding benefit that each one offers to the low latency networks. Finally, one of those is chosen, adding the extra value of the study and the proposed solution to the problem.

7.1.5 Case Study

Once all the models are evaluated, the selected one is studied and simulated against the other alternatives. For it, and taking into account the study necessities, the different scenarios are produced and tested, trying to resemble with real-life situations. Once the performance of the algorithms is evaluated, the cost analysis is done and the thesis study is concluded.

7.1.6 Thesis composition

After the last phase, the documentation related to the project is done. At this last phase, all the previous phases are closed and the documentation of the thesis is written. This information is useful for future works, as well as for other researches that have study the topic before.

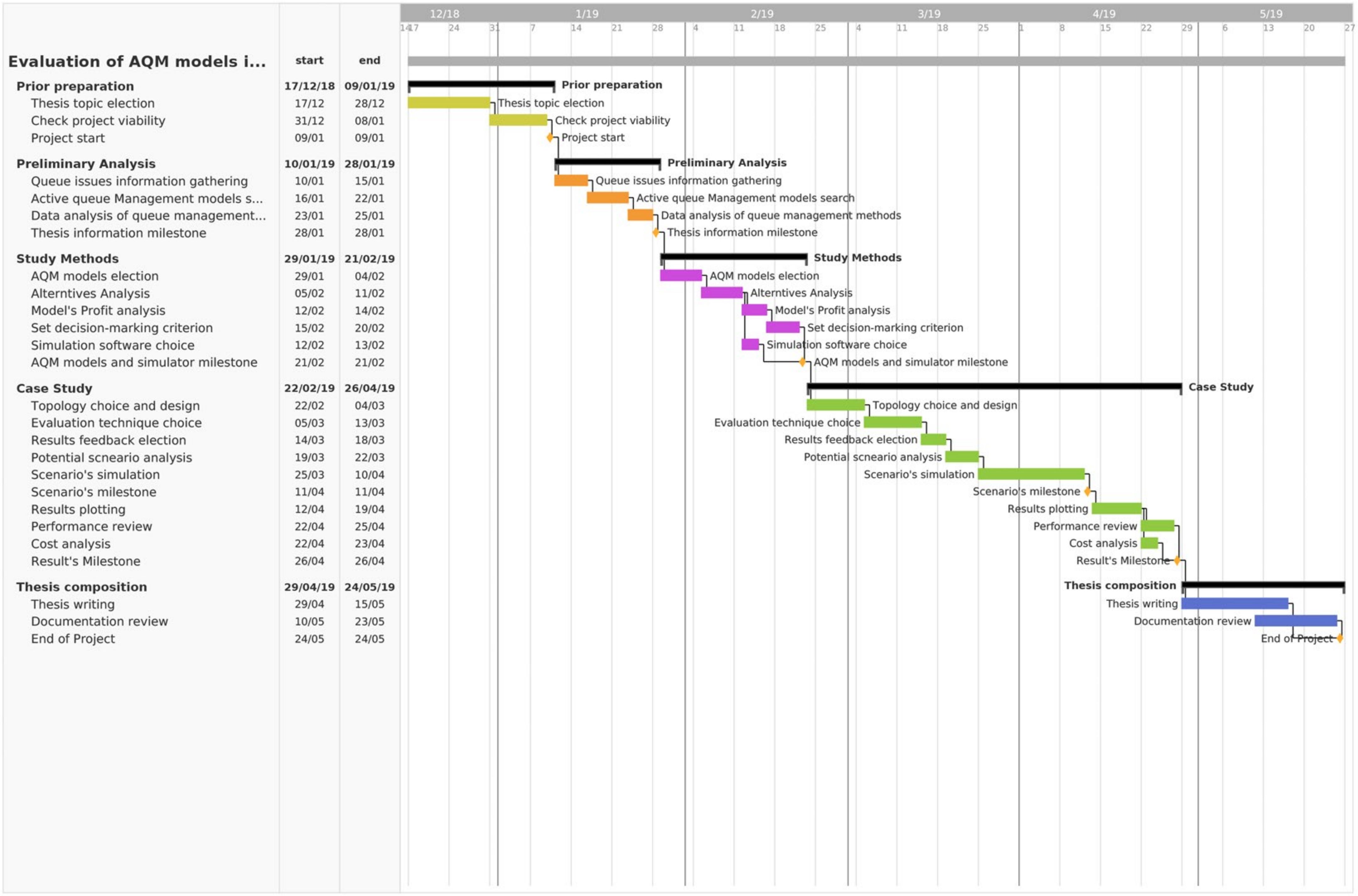
At last, a review of the composition is done, checking and correcting all the weakness of the project, trying to get a totally reliable document.

7.2 Gantt Chart

Page 86 displays Gantt chart of the project. *Evaluation of Alternative Queue Management (AQM) models in Low Latency Networks* project has a duration of 6 months, divided into 5 main tasks. Inside the project, six milestones have been done, checking the progress of the thesis at that point. The chart summarizes the evolution of the project during the time, in a visual and user-friendly way.

Table 10. Work Breakdown Structure

WBS	Task Name	Duration	Start	End
1	**Project Supervision**	**152**	**17/12/2018**	**24/05/2019**
1.1	**Prior Preparation**	**23**	**17/12/2018**	**09/01/2019**
1.1.1	Thesis topic election	11	17/12/2018	28/12/2018
1.1.2	Check project viability	8	31/12/2018	08/01/2019
1.1.3	*Project start*	0	09/01/2019	09/01/2019
1.2	**Preliminary Analysis**	**18**	**10/01/2019**	**28/01/2019**
1.2.1	Queue issues information gathering	5	10/01/2019	15/01/2019
1.2.2	AQM models search	6	16/01/2019	22/01/2019
1.2.3	Data analysis of queue management methods	2	23/01/2019	25/01/2019
1.2.4	*Thesis information milestone*	0	28/01/2019	28/01/2019
1.3	**Study Methods**	**23**	29/01/2019	21/02/2019
1.3.1	AQM models election	6	29/01/2019	04/02/2019
1.3.2	Alternatives analysis	6	05/02/2019	11/02/2019
1.3.3	Model's profit analysis	2	12/02/2019	14/02/2019
1.3.4	Set decision-marking criterion	5	15/02/2019	20/02/2019
1.3.5	Simulation software choice	1	12/02/2019	13/02/2019
1.3.6	*AQM models and simulator milestone*	0	21/02/2019	21/02/2019
1.4	**Case Study**	**63**	**22/02/2019**	**26/04/2019**
1.4.1	Topology choice and design	10	22/02/2019	04/03/2019
1.4.2	Evaluation technique choice	8	05/03/2019	13/03/2019
1.4.3	Results feedback election	4	14/03/2019	18/03/2019
1.4.4	Potential scenario analysis	3	19/03/2019	22/03/2019
1.4.5	Scenario's simulation	16	25/03/2019	10/04/2019
1.4.6	*Scenario's milestone*	0	11/04/2019	11/04/2019
1.4.7	Results plotting	7	12/04/2019	19/04/2019
1.4.8	Performance review	2	22/04/2019	25/04/2019
1.4.9	Cost analysis	1	22/04/2019	23/04/2019
1.4.10	*Result's milestone*	0	26/04/2019	26/04/2019
1.5	**Thesis composition**	**25**	**29/04/2019**	**24/05/2019**
1.5.1	Thesis writing	16	29/04/2019	15/05/2019
1.5.2	Documentation review	13	10/05/2019	23/05/2019
1.5.3	*End of Project*	0	24/05/2019	24/05/2019



Evaluation of AQM models i...	start	end
Prior preparation	**17/12/18**	**09/01/19**
Thesis topic election	17/12	28/12
Check project viability	31/12	08/01
Project start	09/01	09/01
Preliminary Analysis	**10/01/19**	**28/01/19**
Queue issues information gathering	10/01	15/01
Active queue Management models s...	16/01	22/01
Data analysis of queue management...	23/01	25/01
Thesis information milestone	28/01	28/01
Study Methods	**29/01/19**	**21/02/19**
AQM models election	29/01	04/02
Alterntives Analysis	05/02	11/02
Model's Profit analysis	12/02	14/02
Set decision-marking criterion	15/02	20/02
Simulation software choice	12/02	13/02
AQM models and simulator milestone	21/02	21/02
Case Study	**22/02/19**	**26/04/19**
Topology choice and design	22/02	04/03
Evaluation technique choice	05/03	13/03
Results feedback election	14/03	18/03
Potential scneario analysis	19/03	22/03
Scenario's simulation	25/03	10/04
Scenario's milestone	11/04	11/04
Results plotting	12/04	19/04
Performance review	22/04	25/04
Cost analysis	22/04	23/04
Result's Milestone	26/04	26/04
Thesis composition	**29/04/19**	**24/05/19**
Thesis writing	29/04	15/05
Documentation review	10/05	23/05
End of Project	24/05	24/05

8 Economic Aspects

8.1 Budget

The next section deals with the project's cost. It is classified in work hours and repayments, ending with the detailed total cost of the project.

8.1.1 Work hours

Work hours section presents the cost of the engineers involved at the project's development.

Table 11. Work Hours Cost

Name	**Rol**	**Hourly rate**	**Hours**	**Total**
Xabier Sanz	Junior Engineer	20€/h	760 h	15.200€
Eduardas Kutka	Senior Engineer	50€/h	30 h	1.500€
Charles Pinto	Senior Engineer	50€/h	20 h	1000€
				TOTAL: 18.300€

As detailed in Table 10, the research have been done by three engineers. Xabier Sanz, the junior engineer, have developed all the project following the task showed in section 7. On the other hand, senior engineers have supervised the project during the 158 days of work, leading and helping with the development.

Table 12 reveal the cost of each phase, excluding the project supervision done by the superior engineers. The total amount matches with the cost of the junior engineer presented in Table 13.

Table 12. Cost by Phases

Phase	**Supervisor**	**Hours**	**Cost**
Prior Preparation	Junior Engineer	115 h	2.300 €
Preliminary Analysis	Junior Engineer	90 h	1.800 €
Study Methods	Junior Engineer	115 h	2.300 €
Case Study	Junior Engineer	315 h	6.300 €
Thesis composition	Junior Engineer	125 h	2.500 €
			TOTAL: 15.200€

8.1.2 Repayments

The principal repayment of the project is the computer used for the simulations. The computer has process all the simulations with Ubuntu OS, with ns3 free software, not included at project's cost.

Table 13. Work Hours Cost

Item	**Value**	**Remaining value**	**Lifespan**	**Consumption**
Computer	1000 €	166.66 €/year	6 years	6 months
				TOTAL: 83.33 €

8.1.3 Total Cost

The total cost is de sum of the work hours and repayments. In addition, the 8% of indirect cost should be taken into consideration.

Table 14. Total Cost

Item	**Subtotal**
Work hours	18.300 €
Repayments	83.33 €
	SUBTOTAL: 18.383.33€
Indirect costs (8%)	1.470,66 €
	TOTAL: 19.854€

The overall cost of the project is 19.854€, without including VAT costs.

Conclusions and Recommendations

Low latency networks require the modification of the actual queue management for the needs of time-sensitive applications. The evaluation has presented the bufferbloat problem and the AQM methods that correct the issue. Additionally, it has proved how FQ_CoDel, by his own, offers the best delay values, as well as the possibility of the combination with ABE for increasing the throughput performance. All the simulations are done in order to resemble real networks, with real nodes, traffic types, and distances.

FQ_CoDel with ABE offers optimal results for nowadays and future time-sensitive applications. It provides flow isolation and prioritization for new flows, making 'mice' traffic responses quicker and better in time, decreasing delays. Moreover, the use of ABE with explicit congestion notification as sender's modification provides an increase in the throughput utilization. However, it is crucial to persuade network administrators to the activation of ECN, disabled by default in the majority of routers. On the other hand, FQ_CoDel + ABE has some inconvenient with the classification of flows with the use of IPsec or tunnels such VPN. Future works could progress in several ways, one of it is resolving tunnels issues or with not wired networks such as WiFi, where the RTT varies heavily, doing the use of CoDel harder.

All in all, the work shows how AQM models decrease heavily delay times, being the path to follow against bufferbloat. Among the possibles techniques, FQ_CoDel+ABE has been the one selected by the work, having proved his effectiveness both, at the queue and sender. Next steps are focused in protocols as DCTCP and CAKE, whose combine all the benefits of delay decreasing techniques and have the target to being optimal in wireless networks too. However, it is a long way to go in order to be ready for 5G networks and ultra-low latency networks development.

References

[1] Rong Pan Preethi Natarajan Chiara Piglione Mythili Suryanarayana Prabhu Vijay Subramanian Fred Baker and Bill VerSteeg. PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem, 2013.

[2] Belshe. Bandwidth does not matter much.

[3] Bob Briscoe Anna Brunstrom Andreas Petlund David Hayes David Ros Ing-Jyh Tsang Stein Gjessing Gorry Fairhurst Carsten Griwodz and Michael Welzl. Reducing Internet Latency: A Survey of Techniques and Their Merits. *IEEE Communications Surveys and Tutorials*, 18(3):2149--2196, 2014.

[4] BitTorrent Community. BitTorrent: P2p protocol. https://www.bittorrent.com/lang/es/, 2001--2019.

[5] GnuPlot Community. GnuPlot: Graphing utility. http://www.gnuplot.info/, 1986--2019.

[6] NS-3 Community. NS3: Network simulator 3. https://www.nsnam.org/, 2011--2019.

[7] WireShark Community. WireShark: Network sniffer. https://www.wireshark.org/, 1999--2019.

[8] A. Esterhuizen and A.E. Krzesinski. TCP Congestion Control Comparison, 2012.

[9] Eduard Grigorescu Chamil Kulatunga Gorry Fairhurst and Nicolas Kuhn. Evaluation of Priority Scheduling and Flow Starvation for Thin Streams with FQ-CoDel, 2015.

[10] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance, 1993.

[11] Ilya Grigorik. *High Performance Browser Networking*. O'Reilly Media, 2013.

[12] Jianyong Chen Cunying Hu and Zhen Ji. Self-Tuning Random Early Detection Algorithm to Improve Performance of Network Transmission. *Mathematical Problems in Engineering*, 2011(1):0--17, 2011.

[13] M. Allman V. Paxson ICSI and E. Blanton. TCP Congestion Control. RFC 2581, RFC Editor, September 2009.

[14] N. Nichols V. Jacobson. Controlled Delay Active Queue Management. RFC 8289, Internet ENgineering Task Force, January 2018.

[15] Ilpo Jarvinen and Markku Kojo. Evaluating CoDel, PIE, and HRED AQM Techniques with Load Transients, 2014.

[16] Junzhou Luo Jiahui Jin and Feng Shan. Standardization of Low-Latency TCP with Explicit Congestion Notification: A Survey. *IEEE Internet Computing*, 21(1):48--55, 2017.

[17] Joel Hasbroucka and Gideon Saar. Low-latency trading. *Journal of Financial Markets*, 16:646--679, 2013.

[18] Naeem Khademi. *Reducing Latency in Internet Access Links with Mechanisms in Endpoints and within the Network.* PhD dissertation, University of Oslo, 2015.

[19] Nicolas Kuhny and David Rosz. Improving PIE's performance over high-delay paths, 2016.

[20] Naeem Khademi Michael Welzl Grenville Armitage† Chamil Kulatunga. Alternative Backoff: Achieving Low Latency and High Throughput with ECN and AQM, 2015.

[21] Simula Research Laboratory. RITE Project.

[22] Wolfram Lautenschlaeger and Andrea Francini. Global Synchronization Protection for Bandwidth Sharing TCP Flows in High-Speed Links, 2015.

[23] Mohammad Rajiullah. Towards a Low Latency Internet: Understanding and Solutions, 2015.

[24] SallyFloyd RamakrishnaGummadi and ScottShenker. Adaptive RED:An Algorithm for Increasing the Robustness of REDs Active QueueManagement, 2001.

[25] K. Ramakrishnan. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, Internet ENgineering Task Force, September 2001.

[26] M Shreedhar and George Varghese. Efficient Fair Queuing using Deficit Round Robin, 1995.

[27] Jim Theodoras. Ultra Low-Latency Financial Networking, 2010.

[28] N. Khademi M. Welzl. TCP Alternative Backoff with ECN (ABE). RFC 8511, Internet ENgineering Task Force, December 2018.

[29] Greg White. Active queue management algorithms for docsis 3.0, 2013.

[30] Bartek Wydrowski and Moshe Zukerman. On the Transition to a Low Latency TCP/IP Internet, 2002.

Appendices

The appendx A explains how the simulations are done, explaining all the scripts and the command used for the develop.

A PRODUCT SPECIFICATION

Appendix A present the explanation of the script usage at ns3, as well as the default values used at the simulations for the purpose of being recreated by other researchers or users.

A.1 Script usage

The script used for the simulation is called dumbAQM.cc, written in C++ and executed by ns3 simulator. It has to be saved at /scratch folder, and the execution prompt the values of different parameters at the shell. In addition, it generates .pcap files of the bottleneck and AQM queue bytes values per second.

The execution is done with the following command:

```
/ns-allinone-3.29/ns-3.29\$ sudo ./waf --run scratch/dumbAQM
```

It will generate all the scenarios values for the later process of them. If the values want to be saved in a file, Linux echo could be use:

```
/ns-allinone-3.29/ns-3.29\$ sudo ./waf --run scratch/dumbAQM >
    scenario.txt
```

Inside the script, exist different values to be changed:

- **latbot**: It sets the value of the bottleneck latency(i.e. 15ms,30ms,100ms)
- **ratebot**: It sets the bottleneck rate (i.e. 5Mbps)
- **latedg**: It sets edges latency (i.e. 5ms, 10ms, 50ms)
- **reateedg**: It sets edges rate (i.e. 100Mbps)
- **queueDiscType**: It sets the AQM technique (i.e. PfifoFast, Codel or FQCoDel)
- **queueDiscSize**: It sets the queue size, in our case $2 * BDP$ (i.e. 333)
- **simDuration**: It sets the duration of the simulation (seconds).
- **** Application Test ****: The application test section provides the opportunity of changing the throughput, duration, number of packets values. In addition, they could be turned on/off for with the purpose of checking only TCP or UDP behavior.

The plotting have been done with Wireshark's TCP graph option and Gnuplot. Gnuplot default used for plotting have been:

```
set title "TITLE OF THE GRAPH"
set xlabel "X AXIS LABEL"
set ylabel "Y AXIS LABEL"
set terminal pdf
set output "NAME_OF_PLOT_FILE.pdf"
plot 'file' w l title "AQM MODEL", 'file2' w l title "AQM MODEL2",
    ...
exit
```

A.2 Configuration of ns3 parameters

Used ns3 version have been 3.29, with the all in one package provided by the official webpage. The parameters used at simulations are the showed at Table 15:

Table 15. ns3 Parameters

Parameter	Value
ns3 version	3.29
CoDel target	5
Codel interval	100
latbot	15/30/100 ms
ratebot	5Mbps
latedg	5/10/50 ms
rateedg	100Mbps
queueDiscSize	42/84/333
queueSize	100
packet size	1500
simDuration	30