Model Predictive Congestion Control in RDMA-Enabled High-Speed Datacenter Network

by

Yiming Zheng

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Control Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

With the growing demand for online applications such as high-resolution video streaming and cloud gaming, there is an urgent need for high-throughput and low-latency technologies. The Remote Direct Memory Access (RDMA) over Converged Ethernet Version 2 (RoCEv2) network protocol is increasingly preferred in Data Center Networks (DCNs) as it reduces overhead from processor utilization, an issue with the traditional TCP/IP protocol. Along with these technological advancements, effective congestion control (CC) algorithms are vital. Without proper CC, DCNs could experience performance degradation, affecting the Quality-of-Service (QoS). Traditional CCs are not optimized for DCNs as they were developed for Wide Area Networks (WANs). Even though some CCs, such as TIMELY, were designed specifically for DCNs, they struggle with inevitable packet queuing that causes extra latency. High Precision Congestion Control (HPCC) uses in-flight byte estimation for proactive control and performs better than its peers. However, HPCC, like all other CCs, does not employ a mathematical dynamic equation to evaluate network conditions for optimal control actions. This research dives deep into the fundamental dynamics of DCNs and develops a state-space model to explain network behaviors. Leveraging this model, we present the Model Predictive Congestion Control (MPCC). This new method incorporates constrained mathematical optimization and model predictive control (MPC) design principles. As a window-based CC, MPCC adjusts its window size dynamically based on observed network conditions. In experiments ranging from bursty incast to real-world traffic patterns, we show that MPCC significantly reduces switch queue length and flow completion time while maintaining fairness and throughput.

Preface

This thesis is an original work by Yiming Zheng. No part of this thesis has been previously published.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my esteemed supervisors, Dr. Zhan Shu and Dr. Qing Zhao. Their profound professional knowledge not only guided me in tackling academic challenges but also shaped my approach to life itself. Their patience and unwavering support have been instrumental in my personal and academic growth.

I am profoundly thankful to my parents, Dr. Maobo Zheng and Yan Yu, for their invaluable guidance, drawing from their own research experiences. Their emotional and financial support have been the pillars upon which I built my academic pursuits.

I extend my sincere thanks to my colleagues and friends, including Kunlin Zhang, Haoran Qi, Lirui Yu, Xinzhou Gao, Tong Li, Ruiwen Ma, Qiushi Ye, and others. Their constructive academic ideas and insightful suggestions have been instrumental in shaping the direction of my research. Their unwavering support and camaraderie have made this journey more meaningful.

Lastly, I want to express my gratitude for all the difficulties and challenges that I encountered and successfully overcame during this academic journey. These challenges have been valuable learning experiences that have contributed to my personal and academic development.

Each of these individuals and experiences has played a vital role in my academic accomplishments, and for that, I am deeply thankful.

Table of Contents

List of Tables

List of Figures

List of Symbols

Latin

Δt	Discrete interval between two consecutive ACKs.
A_{qp}	matrix that defines the linear transformation of the optimization variable vector for this set of constraints.
D	Round-trip time of a flow path.
g	Gradient vector.
H	Hessian matrix.
lb	Lower bounds on the optimization variable vector.
lbA_{qp}	Lower bounds of the linear matrix inequality constraints.
N	Number of flows that pass an egress queue.
P	Prediction horizon length.
$P_{BD,max}$	Bandwidth-delay product of a flow path.
Q	Queue length of an egress queue.
r_u	Trajectory reference for the input.
r_x	Trajectory reference for the state.
$R_{in,max}$	Maximum data rate of one flow into the switch from a sender.
R_{in}	Data rate of one flow into the switch from a sender.
R_{out}	Data rate of a flow out of the switch egress queue.
u	Control input for general control systems.
ub	Upper bounds on the optimization variable vector.
ubA_{qp}	Upper bounds of the linear matrix inequality constraints.
W	Congestion window size of one sender.
w_0	Manipulated parameter for the dynamics of constraints.

W_Q Positive definite weighting matrices for the state.

W_R Positive definite weighting matrices for the input.

x State variable for general control systems.

x_{qp} Optimization variable vector for solving Quadratic Programming.

Abbreviations

ACK Acknowledgment.

AIMD Additive Increase and Multiplicative Decrease.

AQM Active Queue Management.

BDP Bandwidth-Delay Product.

CC Congestion Control.

CWND Congestion Window.

DCN Data Center Network.

DCTCP Data Center TCP.

DIP Destination IP.

DPORT Destination Port.

FCS Finite Control Set.

FCT Flow Completion Time.

FUSO Multi-Path Loss Recovery.

HoL Head-of-Line.

HPCC High Precision Congestion Control.

INT In-band Network Telemetry.

LAN Local Area Network.

LTI Linear Time-Invariant.

maxBDP Maximum Bandwidth-Delay Product.

MPC Model Predictive Control.

MPCC Model Predictive Congestion Control.

NIC Network Interface Card.

PFC Priority-based Flow Control.

PI Proportional–Integral.

PID Proportional–Integral–Derivative.

QoS Quality-of-Service.

QP Quadratic Programming.

RCC Receiver-Driven RDMA Congestion Control.

RCP Rate Control Protocol.

RDMA Remote Direct Memory Access.

RoCE RDMA over Converged Ethernet.

RTT Round-Trip Time.

S-PFC Selective-Priority-based Flow Control.

SIP Source IP.

SPORT Source Port.

ToR Top-of-Rack.

WAN Wide Area Network.

XCP Explicit Control Protocol.

Chapter 1

Introduction

1.1 Motivation

Due to the increasing demand for online applications, such as video streaming and real-time gaming, and the decreasing price of high-speed Data Center Networks (DCNs) hardware, there has been an ever-growing deployment of data centers. Low latency and high bandwidth become two crucial parameters to determine the Quality-of-Service (QoS) that can keep customers satisfied. For this reason, different companies and researchers continue to increase the link speed even reaching 800 Gbps [1]. Besides the link speed, the traditional software-based network protocol stack, such as TCP/IP stack, can no longer meet the requirement due to the long overhead [2]. To reduce these additional latency resulting from the protocol overhead and processor utilization, Remote Direct Memory Access (RDMA) over Converged Ethernet Version 2 (RoCEv2) is becoming more popular [2–4], but with a fundamental problem. Congestion Control (CC) algorithms specially designed for RDMA-enabled DCNs are sparse. Traditional Additive Increase and Multiplicative Decrease (AIMD) [5] CC algorithms, such as loss-based TCP-CUBIC [6], and delayed-based Fast-TCP [7] are specifically designed for Wide Area Networks (WANs). Their slow convergence speed is unacceptable for high-speed DCNs. Although some CCs are specifically designed for DCNs, such as Data Center TCP (DCTCP) [8] and TIMELY [9], these CCs lack foresight and only involve after-the-switch queue built-up, which will cause extra transmission

latency. In addition, some of the CCs like DCTCP, need heavy parameter tuning. This will increase the complexity for DCN operators and also the risk of incorrect setting of parameters. High Precision Congestion Control (HPCC) [10] is one of the latest CCs designed for RDMA-enabled DCNs. It measures the in-flight byte to gain prediction and dynamically adjust the Congestion Window (CWND) size. Receiver-driven Rapid Congestion Control (RCC) [11] involves a Proportional–Integral (PI) controller to adjust CWND based on the delay as feedback. But RCC and all of its predecessors do not have a rigorous mathematical formulation to show that the DCN parameters like throughput and fairness will converge optimally.

1.2 Literature Review

1.2.1 Congestion Control Types and Related

Since the inception and progression of the Internet, a diverse array of CC algorithms have been developed, ranging from simpler AIMD-based CCs like TCP-Tahoe [12], to more intricate systems such as HPCC [10]. Predominantly, these CC algorithms are implemented at the data sender's end, utilizing various feedback mechanisms from the communication network to determine network conditions. However, certain CCs are implemented at different network nodes: switches, exemplified by Explicit Control Protocol (XCP) [13] and Rate Control Protocol (RCP) [14], and receivers, as seen in RCC [11]. Despite these variations, the underlying principles remain consistent. The upcoming subsection will detail CCs that employ distinct feedback types. Based on these feedback mechanisms, senders calibrate their data transmission amount to the Internet. To mitigate congestion risks, CCs regulate the CWND size, thereby controlling the volume of unacknowledged 'in-flight' packets, or they adjust the sender's pacing rate.

Loss/Drop-Based Congestion Control. Loss-based CCs, such as Tahoe and CUBIC [6, 12], operate by detecting specific signals to detect packet loss in trans-

mission to the receiver. This signal often manifests as a missing Acknowledgement (ACK) for a particular packet. Upon identifying a missed packet, the sender responds by restricting the CWND size. A notable limitation of this approach is that Loss-Based CCs initiate a reduction in sending capacity only after a packet loss occurs, which means congestion has already occurred by the time of detection. Consequently, loss-based CCs lack the foresight to anticipate packet loss.

Alterations have been introduced to loss-based CCs to enable activation prior to packet loss occurrences. A notable method adopted is Active Queue Management (AQM) [15], which requires modifications not to the CCs but to the switches. In AQM, switches are required to proactively monitor and manage their queue lengths, deliberately dropping packets to prevent queues from becoming fully saturated. This transformation converts loss-based CCs into drop-based CCs, like [16]. With the basic mechanism of AQM in place, there has been an integration of more sophisticated control designs, including Proportional–Integral–Derivative (PID) [17] and Model Predictive Control (MPC) controllers [18]. These advanced controllers are utilized in AQM to make packet drop decisions, moving beyond the use of simple threshold-based mechanisms.

Delay-Based Congestion Control. Delay serves as an alternative feedback parameter extensively utilized by CCs. In contrast to drop-based CCs, which require AQM-enabled switches for implementation, CCs utilizing delay as a feedback parameter are capable of responding to congestion proactively, prior to packet loss, and without requiring modifications to switches. Delay, as a measure of congestion, is inherently conveyed with every ACK, reflecting its linear correlation with the queue length in network switches, a concept that will be elaborated upon in the background section. In essence, an increase in delay signals an increasing queue length, indicating severer congestion within the switches. Examples of delay-based CCs, such as TCP-Vegas [19] and FAST [7], demonstrate superior performance compared to loss-based CCs like TCP-Tahoe and TCP-Reno, primarily due to their proactive nature.

However, the challenge in delay-based CCs lies in the accurate measurement of delay for refined control. Even small noisy fluctuations in delay measurement can lead to overreactions or underreactions by the CCs. Google suggests two solutions to this issue [9]: (i) Deploying advanced Network Interface Controllers (NIC) for precise delay measurement, and (ii) Integrating delay in hybrid schemes with other network parameters, such as packet loss. This leads to the subsequent discussion on hybrid-based CCs.

Hybrid-Based Congestion Control. With the evolution of CC algorithms, reliance on a single feedback parameter, such as solely loss or delay-based mechanisms, has shown inadequate for meeting the demands of high-speed and low-latency networks, particularly in DCNs. Traditional CC approaches often result in under utilization and long latency, failing to fully exploit the capabilities of DCNs. A key strategy to overcome these limitations is the development of advanced CCs that leverage multiple parameters, providing a more accurate reflection of network conditions. Implementing such comprehensive solutions is typically challenging in WANs due to concerns related to privacy, security, and hardware compatibility. However, DCNs, with their known network structures including line rate and number of hops, present a unique opportunity.

In DCNs, the availability of detailed structural information and the potential for hardware customization [20] enable the acquisition of a broader range of network parameters. TIMELY [9] is an example of hybrid-based CCs that utilize known line rate information that is intrinsic to the structure of DCNs, combined with precise measurement of timestamps and delay facilitated by customizable NICs. Similarly, HPCC [10] capitalizes on In-band Network Telemetry (INT) technology [21], also made feasible through customizable hardware, to gather comprehensive data, such as the count of switches that a packet traverses and queue lengths at different hops. These innovations reflect a significant shift in CC design, tailored to optimize the performance and potential of DCNs.

Traffic Control Method. In addition to CC algorithms, alternative traffic control methods are deployed to ensure the smooth functioning of Internet networks, particularly when CCs prove inadequate or ineffective in extreme network conditions like incast or Distributed Denial of Service attacks. One such method is Priority-based Flow Control (PFC) [22], which is designed to facilitate a lossless network environment, crucial for the efficient operation of RDMA-enabled DCNs. PFC operates by monitoring switch queue lengths; when the usage exceeds a predefined threshold, PFC sends a PAUSE signal back to the sender to stop transmission until congestion is relieved.

However, PFC can introduce several side effects, such as Head-of-Line (HoL) blocking [3] and PFC storms [10], which can lead to network performance degradation. To mitigate these issues, a variation known as Selective-PFC (S-PFC) [23] has been developed. S-PFC enhances the overall performance of DCNs by selectively allowing certain network areas to experience lossy conditions.

Loss Recovery. Integrating CC algorithms with traffic control methods can substantially reduce network losses, bringing networks closer to a lossless state. Despite these advancements, a completely lossless guarantee is unattainable due to inherent factors like the transmission loss rate in optical fiber cables and the possibility of hardware failures. Therefore, implementing a loss recovery mechanism remains essential, even in RDMA-enabled DCNs.

A widely utilized loss recovery approach in many CCs is the fast retransmit algorithm [12]. It activates when the sender receives three duplicate ACKs, interpreting it as a signal of packet loss and prompting the retransmission of the packet while concurrently reducing the CWND size. However, this method is not ideally suited for DCNs, where losses often result from factors that are unrelated to the CC algorithm or buffer overflow, leading to under utilization of the DCNs' capabilities.

Given these specificities, the development of loss recovery mechanisms tailored for DCNs is crucial. An example of such an innovation is the Fast Multi-Path Loss

Recovery (FUSO) algorithm [24], which is designed to address the unique challenges of DCNs. FUSO immediately redirects recovery packets via an alternate path upon detecting a loss in a flow path, without awaiting the typical timeout duration inherent in traditional fast retransmit algorithms. This approach enhances the efficiency of loss recovery in DCNs, ensuring more effective and timely remediation of packet losses, thereby contributing to the overall performance and reliability of DCNs.

1.2.2 Model Predictive Control in Real-World

In this thesis, MPC is implemented as the CC controller for DCNs. It is also known as receding horizon optimal control, whose versatility and effectiveness not only in control and optimization theory but also in practical applications have been demonstrated. An advantage is that the MPC controller solves the future input optimally given the system dynamics and cost function. Another benefit of implementing MPC is that it can incorporate system delays and constraints into the controller while maintaining mathematical optimality.

Initially, MPC was considered computationally intensive compared to traditional PID controllers due to the necessity of solving an optimization problem for each control input over a predictive horizon. However, with the evolution of high-speed custom processors [25, 26] and advanced optimization techniques [27–29], MPC has increasingly transitioned from being a simulation-only tool to being deployed in actual plant applications [30]. One example is the MPC deployment in AQM mentioned above [18]. The result shows that MPC-based AQM achieves superior performance compared to PID-based AQM.

In autonomous driving, this work [31] presents an MPC control design using kinematic and dynamic vehicle models. The study analyzes the discretization errors of these models, finding that a 200 ms sampling interval performs comparably well to a 100 ms interval while requiring less computing power for the MPC controller. This is validated through experiments using real vehicles as the system plant and processors

as the controller.

MPC has also enhanced performance in power converter applications by leveraging high-speed digital signal processors to apply Finite Control Set Model Predictive Control (FCS-MPC) to power converters [32]. The findings show that FCS-MPC-based converters surpass traditional methods like pulse-width modulation in terms of simplicity, eliminating the need for additional modulation techniques for different control variables. This simplicity and flexibility make FCS-MPC-based converters a more practical choice for power conversion systems.

Furthermore, MPC has extended its reach into health sciences. The Bergman model [33] is used to describe the human glucose-insulin system. A constrained MPC controller is then applied on experiment plants to adjust insulin infusion rates based on feedback from blood glucose levels [34]. The results indicate that the controller effectively maintains blood glucose levels within an acceptable range, even with external disturbances such as meal intake.

1.3 Our Contributions

In this study, we introduce the Model Predictive Congestion Control (MPCC) algorithm, specifically designed for RDMA-enabled DCNs. MPCC's distinctive feature is its utilization of an MPC controller to achieve predictive insights based on a well-constructed mathematical state-space model. This allows it to effectively maintain the queue of switches in DCNs at nearly zero, even under challenging traffic conditions such as incast, burstiness, and real applications traffic patterns while ensuring fairness, throughput, etc., across different flows. Our primary contributions are as follows:

- Derivation of a discrete state-space realization model from the first-principle equation to describe the dynamics between the queue length of the congested switch and the senders' window size.

- Our research delves deep into the DCNs' behaviors. From this, we obtain a dynamic equation that is crucial for accurately computing the delay associated with ACK feedback. By incorporating this delay into our queue length dynamics, more precise DCN behaviors can be captured.

- Formulation of a cost function for MPC controller, along with the identification of constraints for both queue length and the CWND's maximum and minimum limits.

- Building upon the refined DCNs model, which now includes the system delay, we complete the design of the MPC controller. This involves transforming the constrained MPC optimization challenge into a Quadratic Programming (QP) framework. The QP solver, qpOASES [35], is selected for solving the constrained optimization problem in finite horizon.

Experiments were conducted using the NS-3 simulator [36] to validate the accuracy of our discrete state-space model, which describes the dynamics between the queue length and CWND. Our testing suite encompassed diverse traffic patterns in DCNs, from typical scenarios like incast and long-short flows to real-world traffic patterns, specifically web search and Facebook flows, under the dumbbell topology and the three-level fattree topology. The results demonstrate MPCC's superior performance compared to existing CC algorithms, including DCQCN, DCTCP, TIMELY, and HPCC. Notably, MPCC excels in queue length, and flow completion time. Additionally, it remains comparable performance against these established algorithms when assessed on convergence speed metrics.

1.4 Thesis Outline

This thesis is meticulously structured across six chapters to provide an in-depth exploration of our research as follows:

- Chapter 1 introduces the primary challenges and motivations for improving the performance of CCs in high-speed DCNs. Followed by literature reviews that cover the basic ideas of some existing CC algorithms and MPC controller implementations in different research areas. End with a summary of our significant contributions.

- In Chapter 2, essential background knowledge in DCNs, such as delay calculation and INT technology, is introduced.

- Chapter 3 delves into the details of our modelling approach, backed by rigorous verification processes. The modelling of ACK delay is also discussed.

- The design intricacies of the MPC controller are elaborated in Chapter. 4, This includes cost function construction, constraints identification and QP programming formulation.

- In Chapter 5, experimental design and result are shown and discussed for both dumbbell topology and three-level fattree topology under various real-world application traffic loads. The performance of MPCC is also compared with other CCs in different metrics.

- Chapter 6 concludes our thesis, drawing final insights, and highlighting potential directions for future research endeavors.

Chapter 2

Background

2.1 Network as Control Systems

In the context of network systems, senders receive accurate feedback, such as INT information carried by ACK packets. Based on received feedback, including metrics like Round-Trip Time (RTT) and queue length, the sender will adjust either CWND or pacing rate as input for networks to guarantee the networks are in smooth and uncongested condition. The network's response to these inputs is followed by its intrinsic mechanisms and dynamics.

Thus, senders are considered to have the capability to observe specific states and conditions within the network through the feedback. The entire network can be classified as a closed-loop system. In this scenario, as shown in Fig. 2.1, the network operates as the System Plant. Each sender acts as a Controller. The ACK signal acts as the Feedback Element.

2.2 Data Transmission in Modern Communication Networks

In modern packet-switching communication networks, data are predominantly transmitted in the form of packets. A packet is a unit of data containing two main components: a header and a payload. The header contains control information, such as the Source IP (SIP) address, Source Port (SPORT), Destination IP (DIP) address,

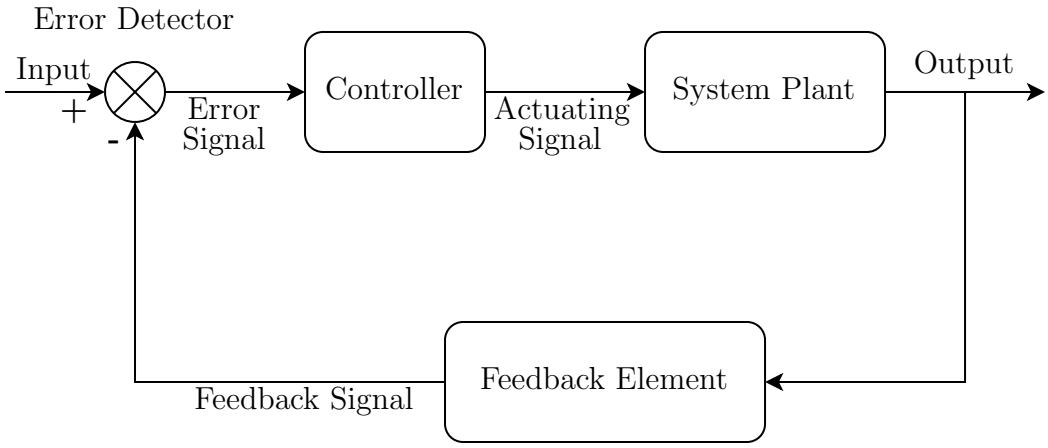


Figure 2.1: A Block Diagram of General Closed-loop Control System

Destination Port (DPORT), and other details like protocol version and the total length of the packet. The payload, on the other hand, carries the actual data being transmitted.

When a packet is dispatched from a sender, it travels through various intermediate switches before reaching its intended destination. Upon reaching a switch, the packet is first received by the ingress queue associated with the incoming link. Based on the routing protocol policy of the switch and the header information of the packet, the switch then determines the specific egress port where the packet is forwarded.

Eventually, the packet arrives at its destination receiver. The receiver, upon successfully accepting the packet, sends an ACK packet back to the corresponding sender. This ACK packet serves to confirm the successful reception of the packet and to request the transmission of the future packet.

Throughout this process, each device in the network, including senders, receivers, and switches, is interconnected via physical links. These links can comprise various transmission mediums, such as optical fiber or coaxial copper wire.

2.3 Undesired Conditions During Packet Transmitting

2.3.1 Physical Linkage and Hardware Failure

Physical linkage and hardware failure are some of the undesired conditions in packet transmission. Various factors, such as signal attenuation and interference, can lead to these failures [37–40]. The primary consequence is the distortion of packet data. Error correction mechanisms like the inclusion of a checksum field in the packet header [41] exist to detect and correct such issues. However, they may not always be effective in extreme cases like multiple-bit errors. In such instances, the network relies on its loss recovery algorithms. Advanced optical fiber technologies and algorithms [42, 43] have reduced physical linkage failures, but the complete elimination of packet loss due to hardware failure remains unattainable.

2.3.2 Improper Network Protocol Setting

Besides hardware issues, software and improper parameter settings can also lead to undesired conditions in network operation. In WANs, due to the limitation in parameter availability, the AIMD-based CUBIC algorithm remains the default CC algorithm in both Linux [44] and Microsoft Windows [45]. CUBIC, as a loss-based CC algorithm, periodically leads to packet drops and reductions in the CWND size. An example of CWND size reduction is shown in Fig. 2.2, which illustrates a packet drop scenario in Windows OS, resulting in throughput fluctuations and a decrease due to CWND reduction. This observation was made during the monitoring of a static download flow. One reason is the increasing number of new joined flows, which can lead to congestion and subsequent packet drops.

In DCNs, more advanced CC algorithms are deployed, making packet drops less frequent. However, even in DCNs, issues can arise if network operators incorrectly set the parameters of these advanced CC algorithms. An example is the DCTCP

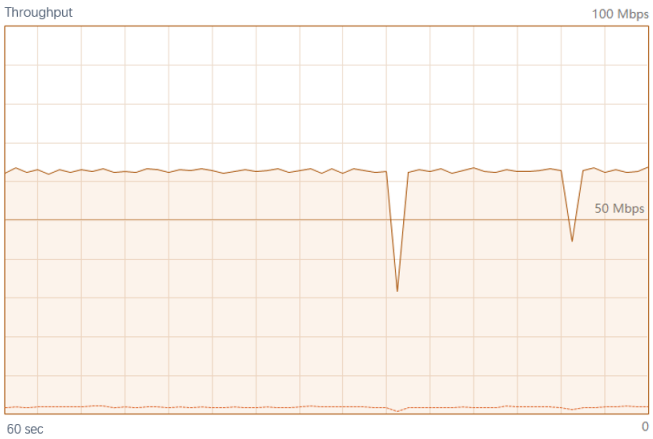


Figure 2.2: A Capture of CWND Reduction in Windows OS

algorithm [8], which involves tuning over 15 parameters. Incorrect configuration of these parameters can adversely affect network performance.

2.3.3 Unexpected Long Delay

Delay is an inherent aspect of packet transmission in communication networks. However, unexpected and prolonged delays can lead to under utilization of the network and a degradation in overall performance. Such delays can severely impact customer experience, evident in scenarios where download speeds fall short of expectations or when significant lag occurs during video chats. In extreme cases, if the delay is excessive, the sender might interpret the packet as lost, thereby initiating the loss recovery mechanism. This reaction leads to the transmission of additional recovery packets, increasing the network's load further. The next section will provide a more detailed discussion on the topic of delays in communication networks.

2.4 Delay in Communication Networks

The delay in communication networks can usually be referred to as RTT. RTT represents the total time from when a sender dispatches a data packet to the moment the sender receives the corresponding ACK signal from the receiver. RTT consists of three main components: link delay, processing delay, and queuing delay.

- Link delay: This is the inherent delay associated with the nature of the transmission linkage medium. For example, in optical fibers, the link delay might be approximately 500 microseconds per 100 kilometers [46]. Link delay is unavoidable and can only be mitigated by using advanced material technologies, not by CC algorithms or protocols.

- Processing delay: This delay refers to the time taken by network devices, such as routers or switches, to process and forward a packet. It is influenced by factors like I/O speed and the complexity of CC algorithms, etc. [47]. Time consumed for header analysis, routing decisions and error checking are also parts of processing delay but usually short. While faster error checking and routing decision protocols can reduce processing delay, the most significant improvements typically come from faster hardware. In this thesis, an estimation of processing delay for each packet is given by the formula:

$$processingDelay = packetSize/linkRate \tag{2.1}$$

- Queue delay: This is the time that a packet spends waiting in a switch queue (buffer) before its transmission. The queue will build up when many packets arrive at the switch simultaneously and the switch cannot process them immediately. Thus the extra packets will be temporarily stored in the queue waiting for forwarding. As the queue size in a switch stacks, the queue delay will be linearly increased. Ideally, the queue delay can be calculated by: $queueDelay = queueLength/linkRate$ In an extreme condition, when the queue is full and cannot accommodate any more packets, switches may drop further incoming packets (drop-tail queue). In this scenario, CCs fail and the networks need to rely on other mechanisms to regulate themselves, such as PFC and loss recovery algorithm. While some switches with AQM may drop packets deliberately before the queue is full to optimize AIMD-based CCs [15], such techniques

are less common in DCNs where advanced CCs have evolved beyond AIMD and AQM.

Therefore, the total RTT can be estimated as:

$$RTT = 2 * \sum_{i=1}^{N} linkDelay + \sum_{i=1}^{N} queueDelay + \sum_{i=1}^{N} processingDelay \tag{2.2}$$

Note that, the 'N' represents the total number of switches a packet traverses. The processing delay for ACK packets is often negligible due to their smaller size compared to data packets. The queue delay for ACK paths is assumed to be zero as the queue length is assumed to be zero.

2.5 Remote Direct Memory Access Technology

RDMA is a technology that enables direct memory access from one computer's memory to others' without involving the processors or operating systems of either computer. Unlike the traditional generalized TCP/IP network protocol stack, which involves significant overhead like copying data between user and kernel spaces, RDMA-enabled networks provide low latency and high throughput. However, deploying RDMA technology in WANs is impractical, as it requires RDMA-capable NICs at both senders and receivers. Conversely, in DCNs, the controlled environment of LANs makes adopting RDMA feasible. As a result, an increasing number of companies are implementing RDMA to enhance their DCN performance [48, 49]. There are primarily two RDMA-enabled protocols:

- RDMA over Converged Ethernet (RoCE) [50]: This protocol enables RDMA over Ethernet networks. RoCEv2, with specific protocol modifications and specialized NICs, brings the benefits of RDMA to Ethernet-based DCNs, combining RDMA's advantages with the widespread use of Ethernet.

- InfiniBand [51]: A high-performance networking technology, InfiniBand is specifically designed for high-performance computing clusters and DCNs, supporting

RDMA natively. Due to its departure from the Ethernet protocol and elimination of Ethernet overhead, InfiniBand networks often outperform those enabled by RoCEv2. However, it requires more specialized hardware and is not compatible with commonly deployed Ethernet protocols. InfiniBand is more often chosen for high-performance computing clusters rather than for typical Ethernet-based DCNs.

2.6 Network Parameter Acquisition

In network CC algorithms, the observation of specific parameters is crucial for effective controlling. Traditional loss-based CC algorithms, such as TCP-CUBIC [6], rely on packet loss as an indicator to trigger CC adjustments, typically modifying the CWND size. In this context, an UNACK signal is sent from receiver to sender after a defined timeout period without receiving an ACK, which signals potential network congestion. Meanwhile, delay-based CCs, such as FAST-TCP [7], use the RTT as a parameter for indicating network congestion. RTT is measured by calculating the time difference between the instance of sending a packet and receiving its corresponding ACK, based on system timestamps.

With the evolution of programmable switches, INT technology [21] is adopted to collect detailed hop-by-hop network status information. These data are pushed to packet headers, enabling senders to receive the network condition information upon the arrival of an ACK. INT technology allows for the acquisition of more precise and fine-grained network parameters, such as the queue lengths at individual switches and timestamps marking a packet's departure from a switch. HPCC [10], for instance, leverages the detailed data provided by INT to surpass other methods in various performance measures, including fast convergence.

DCNs offer additional parameter acquisition opportunities due to their known structures and typologies. These parameters, such as link rate, link delay, and the number of switches that a packet traversed, provide valuable insights. For example,

the link rate can be used to accurately estimate the processing delay of switches. When combined with the known link delay and queue length data from INT technology, RTT can be actively modeled rather than merely measured passively.

2.7 Maximum Bandwidth-Delay Product and Congestion Window Size

The Bandwidth-Delay Product (BDP) is a key metric in network communications, representing the maximum amount of UNACKed packet data, also referred to as in-flight bytes that can exist in the network during transmission between a sender and a receiver. This is similar to the water in a pipe during transmission. Mathematically, BDP is expressed as:

$$BDP = RTT * Rate_{sender}, \tag{2.3}$$

where $Rate_{sender}$ denotes the sender's data transmission rate. The maximum BDP (maxBDP) is determined under conditions where the sender operates at its peak transmission rate and there is only one active flow in the network. MaxBDP can be estimated by the additional parameters provided by the DCN's structure where the sender's rate is known and RTT can be estimated by the eq. (2.2).

In modern RDMA-enabled DCNs, CC algorithms such as HPCC and RCC commonly utilize maxBDP to set both the initial and maximum sizes of the CWND. This approach has significant advantages. It enables the sender to immediately operate at its maximum sending rate at the start of each flow. Even in situations where this new joined flow is competing with existing ones, the CWND size, calculated based on the estimated maxBDP, effectively limits the number of in-flight bytes. This precaution helps prevent the overfilling of switch buffers' egress queues.

This strategy of leveraging maxBDP for CWND sizing is a key factor in the enhanced performance of DCNs compared to WANs. By bypassing the traditional 'Slow

Start' phase found in conventional CC algorithms like TCP-Tahoe [12], DCNs can more rapidly adjust to optimal operating conditions, thereby increasing overall network efficiency and throughput.

Chapter 3

Network Dynamics Modelling

In this chapter, we construct and validate a dynamic model to describe network behavior. Different parameters can indicate network status, such as switch queue length, RTT, throughput, etc. However, switch queue length is a fundamental indicator of the network condition. If the egress queue length at a switch exceeds zero, it implies that the network load surpasses the switch's processing capacity, leading to temporary data storage in the switch's buffer. Conversely, an egress queue length of zero indicates that the switch is coping with the current network load. Importantly, the queue length is a parameter that can be directly transmitted and included in the INT information in a packet header, enabling senders to accurately measure the queue length without observation errors. This enhances the accuracy of mathematical models in network dynamics by utilizing queue length as a key parameter.

3.1 Basic Model

The first principle of continuity, fundamental in any fluid-like system, dictates that the change in content within a system is equal to the difference between the inflow and outflow rates. This principle is exemplified by a system comprising two water pipes connected to a tank: one pipe for inflow and the other for outflow. The change in water content within the tank is the difference between the inflow and outflow rates.

The same concept is similarly applicable to the heating system [52]. Assume $Rate_{in}$ is the rate of heat given to the room and $Rate_{out}$ is the rate of heat loss. Their difference is the rate of heat accumulation in a room. This principle extends to various systems, including traffic signal control and electrical capacitors, and can be generalized in a dynamic equation:

$$\Delta content = \sum Rate_{in} - \sum Rate_{out} \tag{3.1}$$

In DCNs, this principle is observed in the flow of data packets. Similar to fluid systems, data flows in DCNs have rates controlled by senders, and switches buffer queues in the network act as water tanks. The rate of change in the egress queue length at any switch port is determined by the difference between the total data inflow rate and the outflow rate from that port. If the total inflow rate exceeds the outflow rate, the egress queue length increases, and vice versa.

Each egress queue in a DCN is connected to multiple ingress queues but only to one egress link. Therefore, in the context of DCNs, the dynamics of the egress queue can be represented as:

$$\Delta Q = \sum_{i=1}^{N} R_{in} - R_{out}, \tag{3.2}$$

where:

- Q is the queue length at this egress queue.
- N is the number of flows that pass this egress queue.
- R_{in} is the data rate of one flow from a sender.
- R_{out} is the data rate flow out of the egress queue, where the rate depends on conditions that are summarized in Tab. 3.1.

When the queue length is greater than zero, R_{out} corresponds to the link rate, which is the maximum rate of the link connected to the egress port. This occurs

$\sum_{i=1}^{N} R_{in} \geq R_{link}$	$Q > 0$	R_{out}
True	True	R_{link}
True	False	R_{link}
False	True	R_{link}
False	False	$\sum_{i=1}^{N} R_{in}$

Table 3.1: R_{out} Determination Conditions

even if the sum of the inflow rates, $\sum_{i=1}^{N} R_{in} < R_{out}$. This is similar to a water tank with fluid in it, where the maximum outflow rate is determined by the maximum rate capacity of the extraction pipe, regardless of the inflow rate.

If the egress queue length is zero, R_{out} will be the link rate and the queue length starts to build when the $\sum_{i=1}^{N} R_{in} \geq R_{link}$. If the queue length is zero while the $\sum_{i=1}^{N} R_{in} < R_{link}$, then the R_{out} will be $\sum_{i=1}^{N} R_{in}$. This situation is similar to a water tank without fluid, where the outflow rate matches the inflow rate if the inflow is less than the maximum capacity of the extraction pipe. In such cases, the content in the tank remains non-negative. By applying this analogy to DCNs, it implies that when the queue length is zero, the egress link transmits only the data received from the ingress ports, potentially operating below its maximum link rate and leading to under utilization.

To validate the continuity model by eq. (3.2), an experimental setup using a dumbbell topology, as illustrated in Fig. 3.1, is proposed. In this topology, four senders transmit data packets at a consistent link rate of 1 Gbps with 1 μs link delay. Based on the eq. (3.2), the expected queue stacking rate at the egress queue of the bottleneck, switch 1, in this topology is calculated to be 3 Gbps.

To prove this theoretical prediction, a simulation is conducted using NS-3 simulator. The results of this simulation are shown in Fig. 3.2. This figure shows the increasing of the queue length over time at the bottleneck switch 1's egress queue. The simulation results agree with a queue stacking slope of 3 Gbps.

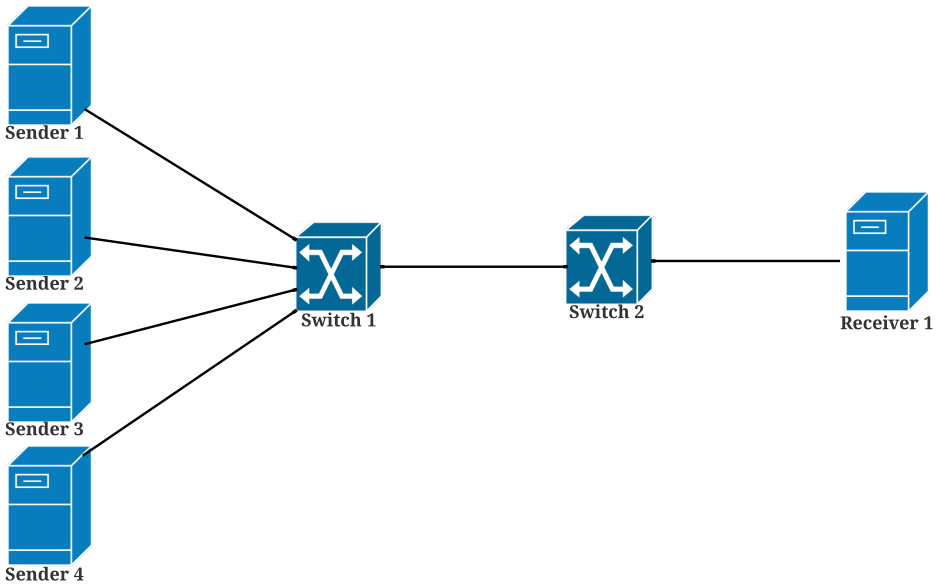


Figure 3.1: An Simplified 4-to-1 Dumbbell Topology

To further evaluate the accuracy and robustness of the model, a series of tests involving step inputs of varying frequencies were conducted. Define a step input with 0 Gbps as the minimum value and 1 Gbps as the maximum value. As illustrated in Fig. 3.3, all four senders transmitted data following the step input function, with frequencies ranging from 100 Hz to 1000 Hz. The observed results showed that both the increasing and decreasing slopes of the queue length were -1 Gbps and 3 Gbps, respectively. This observation is in line with the predictions of the model as expressed in eq. (3.3).

As the frequency of the step input increased, the frequency of the changes in queue length corresponded accurately with the input frequency. This outcome provides substantial validation of the model's correctness and its capability to handle high-frequency inputs effectively. The nature of the network as a discrete model is inherently validated by the fact that input adjustments to the CWND size by the sender occur only upon the receipt of an ACK, rendering considerations of sampling frequency unnecessary.

An additional observation from the experiments was the presence of a flat region at the peak of the queue length. This phenomenon can be attributed to limitations

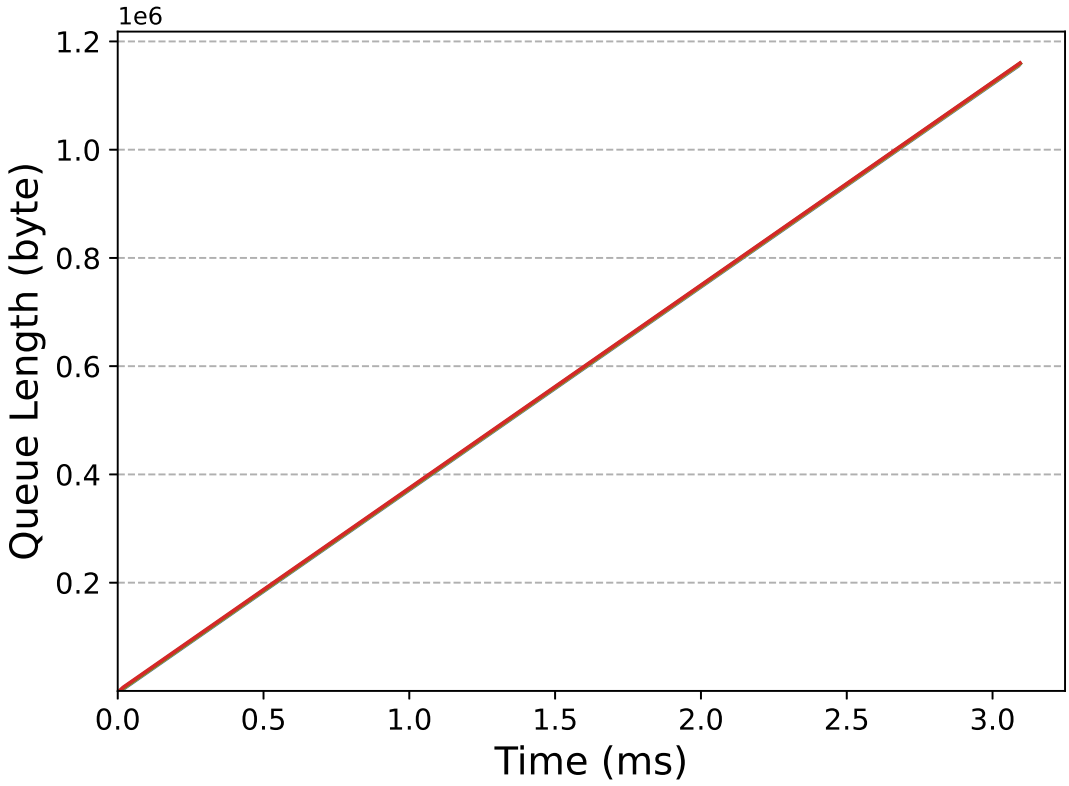


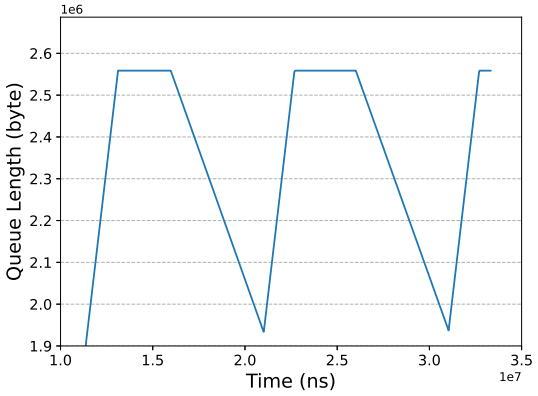
Figure 3.2: An Increasing of Switch Queue Length in 4-to-1 Dumbbell Topology

in the CWND size, which will be discussed in the following section.

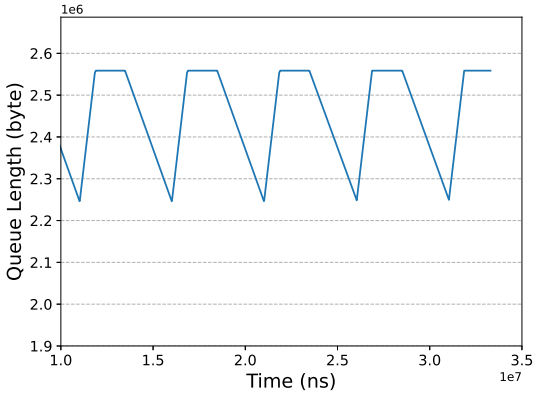
3.2 Model Formulation and Discretization

Although the eq. (3.2) can describe the queue length dynamics, it presents limitations for direct control application due to three primary reasons: (i) The model's continuous nature is not suited for the discrete control actions triggered by the reception of ACK. (ii) The $Rate_{in}$ is not an ideal control parameter when compared to the CWND size, as it cannot prevent switch egress queue overfilling if the CC fails in extreme network conditions. (iii) The queue length is the direct interest rather than $Rate_{in}$, as other parameters, such as RTT, throughput and packet loss possibilities, are directly related to the queue length.

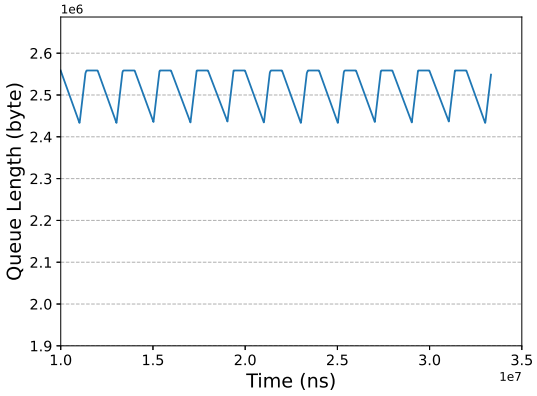
As shown in Fig. 3.4, assume the same topology in Fig. 3.1, the queue length can



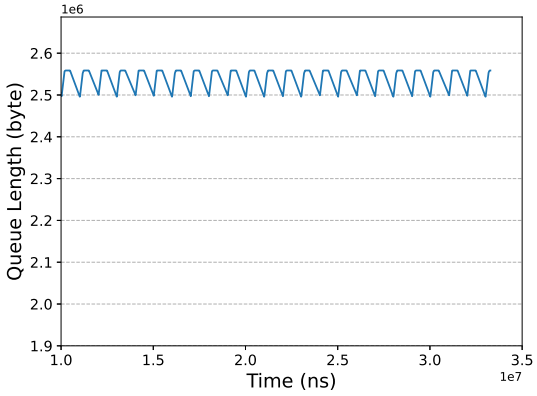
(a) Step Input at 100 Hz



(b) Step Input at 200 Hz



(c) Step Input at 500 Hz



(d) Step Input at 1000 Hz

Figure 3.3: System Response under Step Input with Various Frequency

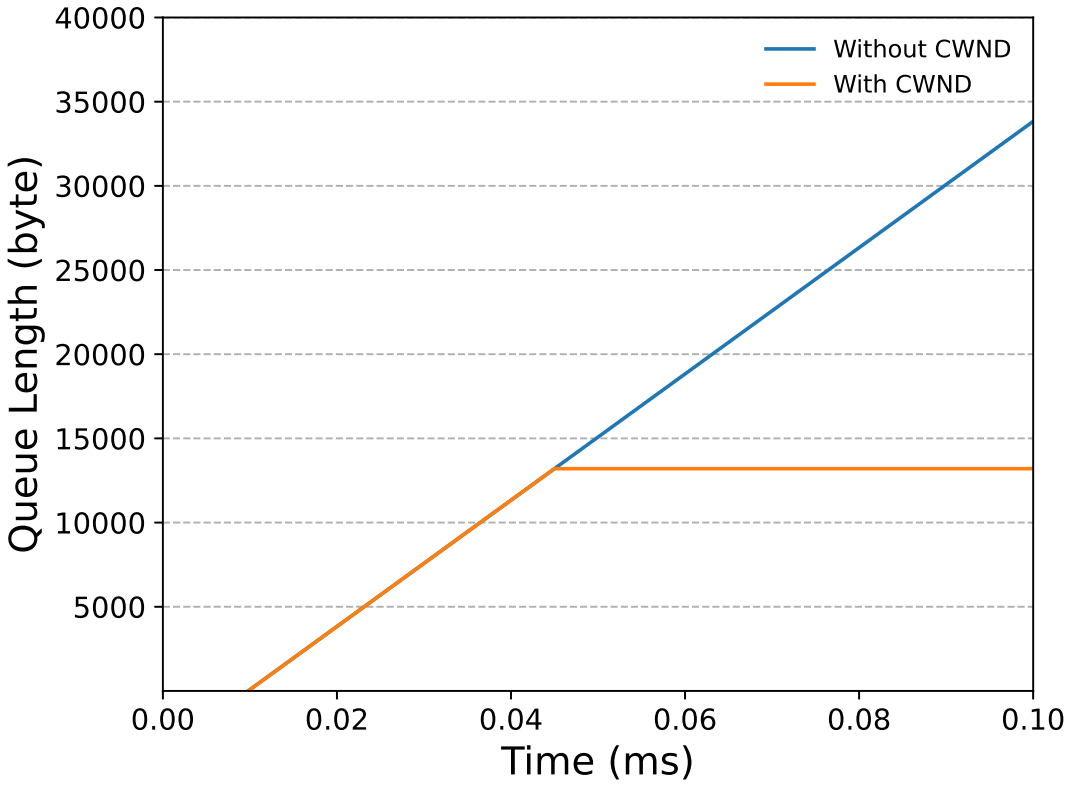


Figure 3.4: The Increasing of Switch Queue Length with CWND vs Without CWND

be effectively bounded by the size of CWND. As discussion in Section 2.7, set BDP, calculable via eq. (2.3), as the CWND size. Set the sender rate, R_{in}, linearly relates with the CWND size as $R_{in} = \frac{R_{in,max}}{P_{BD,max}} * W(k) * \Delta t$, thereby eliminating one extra variable, R_{in}. Under the assumption that all flows have an identical CWND size or rate, a discretized version of eq. (3.2) can be formulated as:

$$Q(k+1) - Q(k) = (\frac{N}{P_{BD,max}} * R_{in,max} * W(k) - R_{out}) * \frac{\Delta t * 1byte}{8bit}, \tag{3.3}$$

- $P_{BD,max}$ is the maximum BDP for this current flow path.
- $R_{in,max}$ is the maximum input rate of each flow.
- Δt is the discrete interval.
- W is the CWND of one sender.

- N is the flow number at the bottleneck egress port.

The flow number (N) can be measured by the switch. This specific flow number measuring method and discrete interval estimation are detailed in Appendix A. After measuring the flow number, the switch incorporates this data into the packet's INT header, which is then conveyed back to the sender via ACK. This mechanism also highlights the challenges in constructing a similar model for WANs due to the absence of INT technology and customizable switches in such environments.

For the purposes of subsequent discussion and control mechanisms to be explored in the following chapter, it is necessary to reformulate eq. (3.3) into a conventional discrete state-space realization form, denoted as $x(k+1) = Ax(k) + Bu(k)$. This leads to a modified version of the equation:

$$Q(k+1) = Q(k) + \frac{\Delta t * N * R_{in,max}}{8 * P_{BD,max}} * (W(k) - \frac{R_{out} * P_{BD,max}}{R_{in,max} * N}) \tag{3.4}$$

3.3 Delayed Model

An additional consideration in the dynamics of DCNs as described by eq. (3.4) is the impact of delay. Fig. 3.2 suggests a minimal influence of delay on the dynamic model, because the queue length increases as soon as the simulation begins. However, an increasing in link delay, such as to 1 ms, significantly shifts this plot to the right.

In the scenario depicted in Fig. 3.5, with a topology and flow setting similar to Fig. 3.1 but with an increased link delay of 1 ms, three distinct plots are presented. The green plot represents the theoretical queue length derived from the delay-free version of eq. (3.4), while the orange plot shows the queue length result captured at switch 1. The blue plot indicates the queue length information carried by the INT header and received by the sender. This figure highlights two types of delays: the input delay, which is the time taken for the CWND input to affect the node, and the feedback delay, which is the time taken for the state information, carried by the ACK's INT header, to reach the sender.

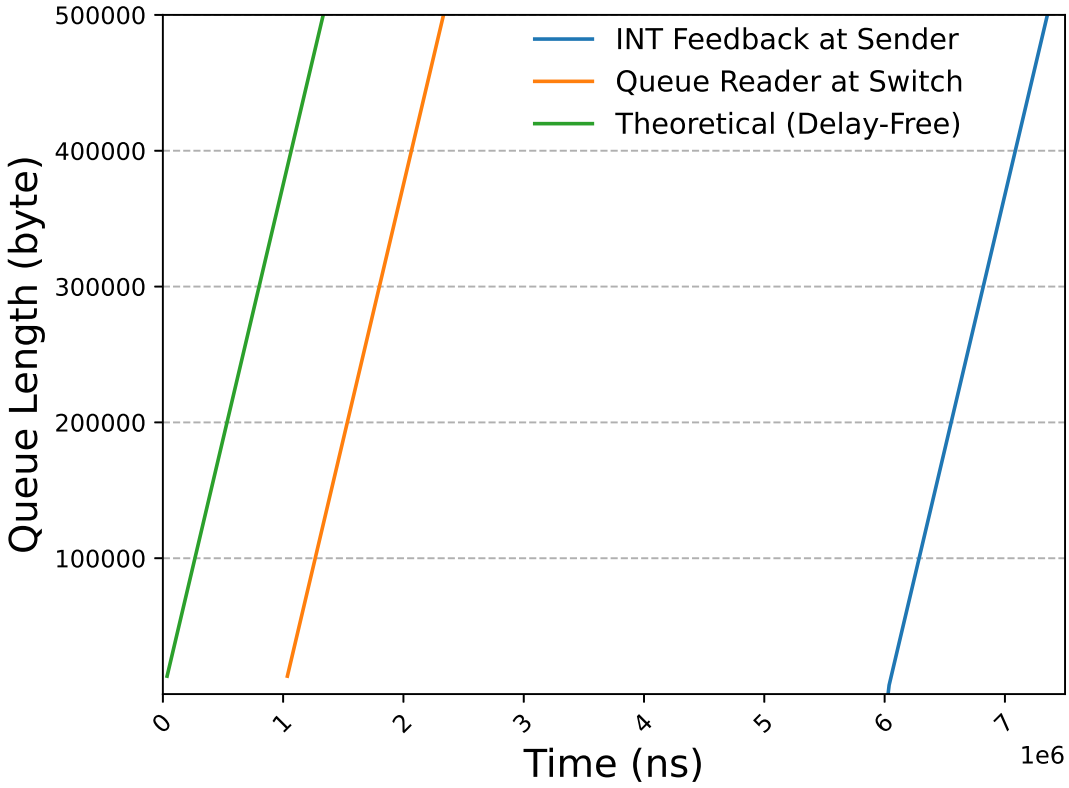


Figure 3.5: The Queue Length of Simulation vs Theoretical Delay-Free Model

Initially, the senders transmit at their link rates. Without delay, an immediate increase in queue length is expected. However, the presence of a 1ms link delay between the senders and switch 1 results in a 1ms delayed increase in egress queue length at switch 1, as shown by the orange plot shifting. Furthermore, the feedback delay, encompassing five additional link traversals for the ACK to reach the sender, causes the sender to recognize the queue length at switch 1 with a delay of 6ms. Here, the processing delay is considered negligible compared to the enlarged link delay. The queue delay will be added to the input delay and feedback delay based on the queue length and the location of switches.

To incorporate the input delay and feedback delay into the model represented by eq. (3.4), discrete delay steps τ_1 and τ_2 are introduced. τ_1 is the discrete input delay steps that calculated by $\tau_1 = \frac{inputDelay}{\Delta t}$ and τ_2 is the discrete feedback delay steps that

calculated by $\tau_2 = \frac{feedbackDelay}{\Delta t}$. Considering DCNs as Linear Time-Invariant (LTI) systems, the discrete state-space realization will be modified to:

$$Q(k+1) = Q(k) + \frac{\Delta t * N * R_{in,max}}{8 * P_{BD,max}} * (W(k - \tau_1 - \tau_2) - \frac{R_{out} * P_{BD,max}}{R_{in,max} * N}) \quad (3.5)$$

Then let:

- $A = I$
- $B = \frac{\Delta t * N * R_{in,max}}{8 * P_{BD,max}}$
- $\overline{W(k)} = (W(k) - \frac{R_{out} * P_{BD,max}}{R_{in,max} * N})$

For practical control application, this state-space model with delay represented by eq. (3.5) needs further manipulations. By defining:

$$\overline{Q} \triangleq Q(k + \tau_1 + \tau_2)$$

Then the state-space model in eq. (3.5) is written as,

$$\overline{Q(k+1)} = A\overline{Q(k)} + B\overline{W(k)} \quad (3.6)$$

where:

$$\overline{Q(k)} = Q(k + \tau_1 + \tau_2) = A^{\tau_1+\tau_2}Q(k) + \sum_{i=0}^{\tau_1+\tau_2-1} A^i B\overline{W(k-1-i)} \quad (3.7)$$

Then, the delayed model is shifted to a delayed-free model by leveraging LTI system characteristics. It requires the system to memorize the past $\tau_1 + \tau_2$ time step input to predict the state and queue length, at the instance where the current input impacts the future.

3.4 State Switch Selection

In the previously discussed section, the focus was on modelling the queue length dynamics of a single switch in DCNs. However, in a network with multiple switches,

attention is primarily directed towards the bottleneck switch, which is key for effective CC. When ACKs received by senders include INT information detailing the queue lengths across all switches, only those switches with queue lengths greater than zero are considered bottlenecks. This is because a non-zero queue length indicates that the rate of incoming data exceeds the switch's forwarding capacity, signifying congestion. Consequently, in the state-space model described in eq. (3.5), the queue length of the bottleneck switch is used as the state variable to facilitate MPCC.

If the queue lengths in all the switches traversed by a packet are zero, this suggests an absence of congestion in the current network conditions. In such cases, the MPCC will maintain the existing size of the CWND. In the context of the 4-to-1 dumbbell topology, as illustrated in Fig. 3.1, it is understood that congestion is most likely to occur at the specific switch, n4. Therefore, the modelling and control strategies focus on monitoring and managing the queue dynamics at this critical bottleneck point to maintain efficient network operation and prevent congestion.

Incast is a phenomenon commonly observed in DCNs, typically caused when multiple data flows arrive simultaneously at the same egress port of a switch. This phenomenon is the most common congestion caused in data center [53], which will result in one-hop congestion. In such scenarios, modelling the network dynamics based on the bottleneck queue effectively captures the essence of one-hop congestion induced by incast.

However, multi-bottleneck congestion can also occur in DCNs. In Fig. 3.6, a situation is illustrated where just three data flows can cause both switch 1 and switch 3 to become bottlenecks. This results in the INT information indicating two queues with lengths greater than zero. While one might consider expanding the state matrix dimension in the state-space equation to accommodate multiple bottlenecks, this approach may not be advantageous given the characteristics of the network.

When the CWND size is reduced to manage congestion, this limits the number of in-flight bytes in the network. Data flows in a network have a temporal characteristic;

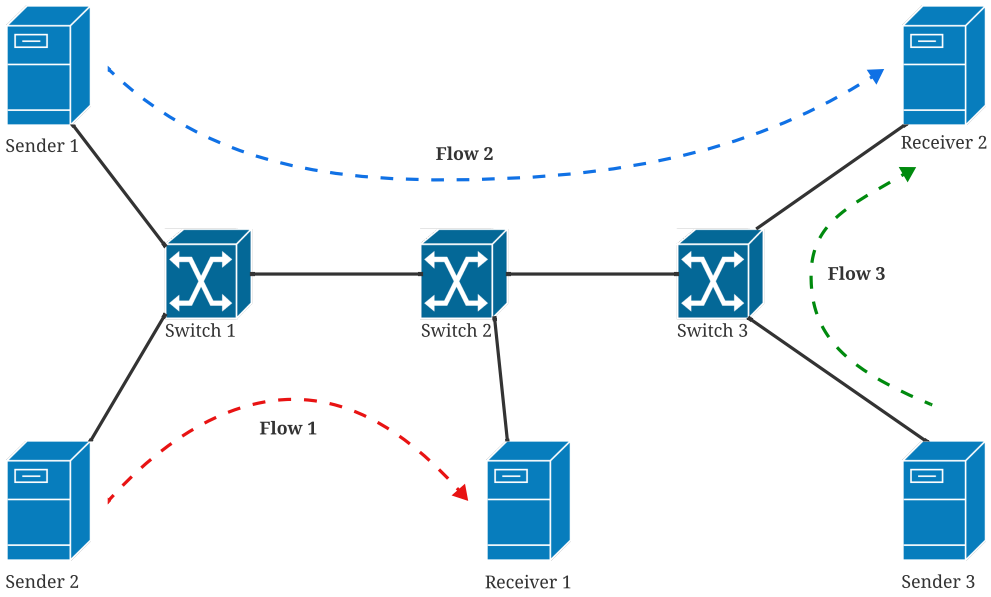


Figure 3.6: Topology and Flows that Cause Multi-hop Congestion

a reduced CWND size will sequentially decrease the queue length at each switch, starting from the one closest to the sender. Expanding the state dimension, while keeping the CWND input dimension as one (since only one CWND can be adjusted per flow), could lead to an excessive reduction in CWND. This would potentially cause under utilization at the first bottleneck switch while leaving the queue length at subsequent switches unchanged.

Therefore, in cases of multi-hop congestion, the most appropriate strategy is to select the most congested queue length as the state for the model. This approach involves adjusting the CWND size over multiple ACK steps to gradually reduce the queue length back to zero, ensuring effective congestion management across multiple bottleneck points in the network while preventing the under utilization of network resources.

The state selection algorithm, outlined in Algorithm 1. All the egress queue length and flow numbers of each individual switch that the current packet transmitted are memorized and carried back by INT header information in the corresponding ACK.

When a sender receives an ACK, it uses this algorithm to determine the most

congested switch, which will then be used as the reference state in the state-space model for congestion control, as defined in eq. (3.6). The algorithm performs a linear comparison among the egress queue lengths (`Q, index`) of different switches that the packet has traversed. The switch with the maximum egress queue length is selected as the state switch for the model. This chosen queue length, along with its associated flow numbers, becomes the state and parameters for the state-space realization represented by eq. (3.4).

This process not only identifies the most congested switch but also helps in determining network parameters such as the link rate, R_{link}, and the maximum Bandwidth-Delay Product, $P_{BD,max}$, by combining topology structure and number of past switches (nhop).

Algorithm 1 State Selection between Switches Egress Queues

```
1: if Receive an ACK then
2:    Q = 0
3:    N = 1
4:    State switch index = 0
5:    for hop,index ≤ nhop do
6:       if Q_index > Q then
7:          Q = Q_index
8:          N = N_index
9:          State switch index = hop,index
10:      end if
11:   end for
12: end if
```

Chapter 4

Control Strategy

4.1 Controller Selection

Many CC algorithms, including TCP-CUBIC and HPCC, rely on heuristic methods to adjust the CWND size. These approaches aim to optimize bandwidth utilization by increasing CWND size when the network is underutilized, and decreasing it when congestion is detected. However, such heuristic methods do not guarantee mathematical optimality and often require many parameters tuning by DCN operators.

RCC embraces a PID controller, a classical control approach. However, PID controllers lack foresight, as they cannot process the intrinsic delay in networks. This limitation can lead to overreactions or underreactions to network conditions. Although RCC attempts to mitigate this effect with a hyperbolic tangent (tanh) function to prevent dramatic CWND adjustments, the choice of this function is not grounded in rigorous mathematical rationale.

The constructed model from the previous chapter paves the way for a model-based approach to congestion control. MPC controller stands out as a suitable choice for CC in DCNs. MPC leverages the concept of receding horizon optimization to manage model inaccuracies and noise effectively. Furthermore, our delayed discrete state-space model, which incorporates various network delays, addresses the issues of overreaction and underreaction typically encountered in PID controllers. Additionally, MPC is capable of managing constraints, such as network bandwidth limitations.

4.2 Observability and Controllability Analysis

The delayed model, as defined by eq. (3.5), is characterized by a constant, 1, for A matrix, and a constant, $\frac{\Delta t * N * R_{in,max}}{8 * P_{BD,max}}$, for B matrix. The state in this model is the queue length, which is the primary measurement obtained from the INT header in ACKs. Since the state (queue length) is the direct measurement of interest, there is no explicit need for an output equation of the form $y(k) = Cx(k) + Du(k)$. In this context, the C matrix can be considered as a constant, 1, and the D matrix as a constant, 0.

Therefore, for system observability and controllability analysis, this scalar state-space equation has a full-rank controllability matrix and a full-rank observability matrix, showing that the system model by eq. (3.5) is controllable and observable.

4.3 Constraints Formulation

Another advantage of MPC is that it can embed different constraints. Here two most important constraints are state (queue length) constraints and control input (CWND) constraints.

4.3.1 Control Input Constraints

Lower Bound of CWND: The CWND size must be greater or equal to zero. A CWND size of zero indicates the absence of in-flight bytes and UNACK packets in the network, resulting in zero throughput. In modern communication networks, throughput cannot be negative as data transmission is unidirectional. The sender cannot reduce throughput below zero to request data packets from the receiver. Instead, if the sender requires data from the receiver, it must send a request, prompting the receiver to initiate a new data flow. In this new flow, the roles are reversed: the sender becomes the receiver and vice versa. Thus, zero is established as the minimum boundary for CWND, serving as the lower constraint for control input. Mathematically, this is

represented as $W_{min} \geq 0$.

Upper Bound of CWND: Theoretically, the CWND size can be set to infinity. However, allowing infinite numbers of UNACKed packets is infeasible and ridiculous. This setting will make the CWND size meaningless because the link rate of the first link, where the sender connects, becomes the limitation and bound. The CWND size then becomes redundant. In practice, setting the CWND size to infinite could lead to long integer overflow. Thus, the maximum CWND size is limited by the data type used to store its value. Nevertheless, this CWND size is still large enough to become insignificant and still bound by the first encountered link rate.

A conventional method, introduced in the background section, links CWND size with the maxBDP, which is determined by the sending rate and network delays. This approach ensures that the network is always bounded by the CWND, maintaining a linear relationship between these two variables. It also ensures efficient throughput performance while preventing queue buildup severely. Consequently, the upper bound of CWND is determined to be the maxBDP of the network, mathematically represented as $W_{max} \leq P_{BD,max}$.

Notice that the input in eq. (3.5) is $(W(k-\tau_1-\tau_2) - \frac{R_{out}*P_{BD,max}}{R_{in,max}*N})$. So we can apply linear shifting to the input and make

$$\overline{W(k-\tau_1-\tau_2)} = (W(k-\tau_1-\tau_2) - \frac{R_{out}*P_{BD,max}}{R_{in,max}*N}) \tag{4.1}$$

for calculation simplification. Therefore, the lower bound constraints for shifted input is $\overline{W_{min}} = -\frac{R_{out}*P_{BD,max}}{R_{in,max}*N}$. The upper bound constraints for shifted input is $\overline{W_{max}} = P_{BD,max} - \frac{R_{out}*P_{BD,max}}{R_{in,max}*N}$. In summary, the input constraint is:

$$\frac{-R_{out}*P_{BD,max}}{R_{in,max}*N} \leq \overline{W} \leq P_{BD,max} - \frac{R_{out}*P_{BD,max}}{R_{in,max}*N} \tag{4.2}$$

4.3.2 State Variable Constraints

Lower Bound of queue length: The lower bound of state (queue length) constraint refers to the minimum allowed number of unprocessed bytes of data temporarily

stored in the switch's egress queue. Similar to the lower bound of the control input (CWND), the queue length is always equal to or greater than zero. A queue length of zero at a specific switch node indicates that the system's state dynamics are in equilibrium, with no congestion detected for this switch. Conversely, a queue length greater than zero signifies congestion at this switch node. The egress queue length cannot be negative, as it represents a scalar unit of data storage, which operates non-directional. Thus, the lower bound of the state variable, egress queue length, is defined as being equal to or greater than zero, mathematically represented as $QLen <= 0$.

Upper Bound of queue length: Theoretically, the upper limit of an individual egress queue's length is determined by the switch's hardware memory capacity. For example, the Cisco Catalyst 9300 Series switches [54] support up to 64 MB of buffer size for queue length. In scenarios where the queue is full, any additional unprocessed packets will be dropped, leading to buffer overflow. To mitigate this, setting an upper bound constraint less than the hardware capacity on the queue length can increase the system's tolerance and prevent overflow. Delving further into the analysis, especially within the context of DCNs topology, consider a scenario with four flows originating from each sender, as depicted in Fig. 3.1. For this topology, congestion is likely at switch 1. The initial CWND size, as per the RoCEv2 protocol, is set to the maxBDP of the path of the current flow. This initial CWND ensures that the sender's initial rate is at the line rate, keeping the queue length at the egress port empty if only the current flow is active. Assuming the CC algorithm is disabled and all senders maintain a constant CWND size equal to maxBDP, only one maxBDP size of in-flight bytes can be accommodated by the network. Therefore, the excess in-flight bytes will accumulate in the queue at switch 1. If all four senders transmit at a line rate with a CWND size equal to maxBDP, the resultant queue length at switch 1 will be $(4-1) * P_{BD,max}$.

In summary, the state constraint is:

$$0 \leq Q(k) \leq (N-1) * P_{BD,max} \tag{4.3}$$

4.4 Cost Function Construction

A cost function represents the objective that MPC controllers aim to optimize. To construct this cost function, we use the following general form:

$$\underset{u}{Min}\ J = \sum_{k=0}^{P-1} \big(x(k) - r_x)^T W_Q (x(k) - r_x) + (u(k) - r_u)^T W_R (u(k) - r_u)\big) + (x(N) - r_x)^T P_t (x(N) - r_x) \tag{4.4}$$

where:

- $x(k)$ is the state at time step k.
- $u(k)$ is the control input at time step k.
- W_Q and W_R are positive definite weighting matrices for the state and control input, respectively.
- P is the prediction horizon.
- P_t is the terminal cost weighting matrix.
- r_x and r_u are the trajectory reference for state and input, respectively.

The first step involves identifying the desired state reference trajectory, which guides the controller toward the target state. In our case, the objective is to minimize the egress queue length at the congested switch towards zero while maintaining the CWND at the fair share size. This approach ensures fairness among flows passing through a node in congestion. Consequently, we set the state trajectory reference, r_x, in eq. (4.4) to zero. At equilibrium, the CWND size is $\frac{R_{out} * P_{BD,max}}{R_{in,max} * N}$, representing the

maximum CWND size that maintains a constant queue length. This fair share CWND not only fully utilizes the network bandwidth but also serves as the input reference for the controller. Hence, the input trajectory, r_u, should be $\frac{R_{out}*P_{BD,max}}{R_{in,max}*N}$. Furthermore, as discussed in the previous section, we used $\overline{W(k)} = (W(k) - \frac{R_{out}*P_{BD,max}}{R_{in,max}*N})$, as the shifted input for computational convenience, leading to $(u(k) - r_u) = \overline{W(k)}$. Therefore, by naming $x(k)$ as $Q(k)$, we reformulate the cost function as:

$$\underset{W}{Min}\ J = \sum_{k=0}^{P-1} \left(Q(k)^T W_Q Q(k) + \overline{W(k)}^T W_R \overline{W(k)} \right) \tag{4.5}$$

Note that the terminal cost is not a crucial consideration in the context of the MPC controller's implementation within the MPCC framework. This is attributed to the receding horizon nature of the controller's operation. Consequently, the terminal cost term is omitted to conserve computational resources and to simplify the process of tuning the weight matrices.

4.5 Quadratic Programming for MPC Solving

4.5.1 Cost Function Quadratic Programming Formulation

Having established the discrete state-space model and the corresponding cost function, we can now implement the MPC controller to compute the controlled input, specifically the CWND size. Our system dynamics are modeled as an LTI discrete state-space model, and we employ a linear cost function alongside linear constraints. Consequently, the task of determining the optimal control input for the MPC controller can be formulated as a QP problem. For solving this QP problem, we utilize an open-source library, qpOASES [35] as the QP solver in this context. According to the qpOASES user manual, a QP problem is required to be structured in the following

form:

$$\min \quad \frac{1}{2} x_{qp}^T H x_{qp} + x_{qp}^T g(w_0) \tag{4.6}$$

$$\text{s.t.} \quad lbA_{qp}(w_0) \leq A_{qp} x_{qp} \leq ubA_{qp}(w_0), \tag{4.7}$$

$$lb(w_0) \leq x_{qp} \leq ub(w_0), \tag{4.8}$$

where:

- x_{qp} is the the optimization variable vector.
- H is the Hessian matrix is symmetric and positive (semi-)definite.
- g is the gradient vector.
- w_0 is the manipulated parameter for the dynamics of constraints.
- A_{qp} This is a matrix that defines the linear transformation of the optimization variable vector for this set of constraints.
- lbA_{qp} and ubA_{qp} are the lower and upper bounds of the linear matrix inequality constraints respectively.
- lb and ub are the lower and upper bounds on the optimization variable vector x_{qp} respectively

Given that the constraints for queue length and CWND size are constants, the w_0 can be omitted.

Next, as an example, we formulate the delay-free state-space model given by eq. (3.4) and integrate the cost function given by eq. (4.5) along with constraints given by eq. (4.2) and eq. (4.3) into the QP problem form that is required by the qpOASES library. The procedure is the same for the delayed state-space model given by eq. (3.5), except the delayed equation needs to be shifted to eq. (3.6), first. At time step k, the control input sequence for P step horizon can be represented as an input

vector with a dimension of P by 1 due to its a single input system, where P denotes the length of the prediction horizon:

$$U_k = \begin{bmatrix} \overline{W(k|k)} \\ \overline{W(k+1|k)} \\ \overline{W(k+2|k)} \\ \cdots \\ \overline{W(k+P-1|k)} \end{bmatrix}$$

At time step k, for a single state system, the sequence of states over a P-step prediction horizon can be encapsulated in a state vector. This vector has dimensions of P by 1, where P represents the length of the prediction horizon:

$$X_k = \begin{bmatrix} Q(k+1|k) \\ Q(k+2|k) \\ Q(k+3|k) \\ \cdots \\ Q(k+P|k) \end{bmatrix}; x(k) = Q(k)$$

And for each element in U_k and X_k, the relationship can be expressed explicitly as:

$$\begin{aligned}
Q(k+1|k) &= A * Q(k) + B * \overline{W(k|k)} \\
Q(k+2|k) &= A^2 * Q(k) + AB * \overline{W(k|k)} + B * \overline{W(k+1|k)} \\
Q(k+3|k) &= A^3 * Q(k) + A^2B * \overline{W(k|k)} + AB * \overline{W(k+1|k)} + B * \overline{W(k+2|k)} \\
&\vdots \\
Q(k+p|k) &= A^p * Q(k) + \sum_{i=0}^{p-1} A^{p-1-i} B * \overline{W(k+i|k)}
\end{aligned}$$

Therefore, the state vector, X_k, and input vector, U_k, can be written as:

$$X_k = \bar{A} * x(k) + \bar{B} * U_k, where : \bar{A} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^p \end{bmatrix}; \bar{B} = \begin{bmatrix} B & 0 & 0 & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ A^2B & AB & B & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A^{p-1}B & A^{p-2}B & A^{p-3}B & \cdots & B \end{bmatrix} \tag{4.9}$$

Then, the cost function is given by eq. (4.5), can be written as:

$$J(U_k) = X_k^T \bar{W}_Q X_k + U_k^T \bar{W}_R U_k, \tag{4.10}$$

where:

$$\bar{W}_Q = \begin{bmatrix} W_Q & 0 & 0 & \cdots & 0 \\ 0 & W_Q & 0 & \cdots & 0 \\ 0 & 0 & W_Q & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & W_Q \end{bmatrix} \bar{W}_R = \begin{bmatrix} W_R & 0 & 0 & \cdots & 0 \\ 0 & W_R & 0 & \cdots & 0 \\ 0 & 0 & W_R & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & W_R \end{bmatrix} \tag{4.11}$$

By substituting eq. (4.9) into eq. (4.10), can get:

$$J(U_k) = (\bar{A}x(k) + \bar{B}U_k)^T \bar{W}_Q (\bar{A}x(k) + \bar{B}U_k) + U_k^T \bar{W}_R U_k$$

$$J(U_k) = U_k^T (\bar{B}^T \bar{W}_Q \bar{B} + \bar{W}_R) U_k + (2(\bar{A}x(k))^T) \bar{W}_Q \bar{B} U_k + (\bar{A}x(k))^T \bar{W}_Q \bar{A}x(k)$$

Now the cost function is given by eq. (4.5), is formulated in eq. (4.10) that agrees to the standard QP form given by eq. (4.6), where:

$$H = 2(\bar{B}^T \bar{W}_Q \bar{B} + \bar{W}_R) \tag{4.12}$$

$$g = 2(\bar{A}x(k)^T) \bar{W}_Q \bar{B})^T \tag{4.13}$$

$$x_{qp} = U_k \tag{4.14}$$

The term, $(\bar{A} * x(k))^T \bar{W_Q} \bar{A} * x(k)$, can be ignored when minimizing for U_k because it only depends on the initial condition, $x(k)$, at step k with constant $\bar{A}$ and $\bar{W_Q}$.

4.5.2 Constrains Quadratic Programming Formulation

The term x_{qp} is the the optimization variable vector that contains every control input, CWND, for the number of prediction horizons in the future, because the QP problem minimizes eq. (4.6), by finding optimal control input series in the finite horizon. For each element in the input vector, x_{qp}, shall obey the constraints in eq. (4.2). Therefore,

$$lb = \begin{bmatrix} \frac{-R_{out} * P_{BD,max}}{R_{in,max} * N} \\ \frac{-R_{out} * P_{BD,max}}{R_{in,max} * N} \\ \frac{-R_{out} * P_{BD,max}}{R_{in,max} * N} \\ \vdots \\ \frac{-R_{out} * P_{BD,max}}{R_{in,max} * N} \end{bmatrix}; ub = \begin{bmatrix} P_{BD,max} - \frac{R_{out} * P_{BD,max}}{R_{in,max} * N} \\ P_{BD,max} - \frac{R_{out} * P_{BD,max}}{R_{in,max} * N} \\ P_{BD,max} - \frac{R_{out} * P_{BD,max}}{R_{in,max} * N} \\ \vdots \\ P_{BD,max} - \frac{R_{out} * P_{BD,max}}{R_{in,max} * N} \end{bmatrix} \tag{4.15}$$

In eq. (4.9), $X_k = \bar{A} * x(k) + \bar{B} * U_k$ can be rewrite to $X_k - \bar{A} * x(k) = \bar{B} * U_k$.By letting $A_{qp} = \bar{B}$, then lbA_{qp} and ubA_{qp} will be:

$$lbA_{qp} \leq X_k - \bar{A} * x(k) \leq ubA_{qp} \tag{4.16}$$

X_k is the state vector in which each element inside shall follow the constraint in eq. (4.3). Therefore,

$$lbA_{qp} = \begin{bmatrix} -Ax(k) \\ -A^2x(k) \\ -A^3x(k) \\ \vdots \\ -A^px(k) \end{bmatrix}; ubA_{qp} = \begin{bmatrix} (N-1) * P_{BD,occupied} - Ax(k) \\ (N-1) * P_{BD,occupied} - A^2x(k) \\ (N-1) * P_{BD,occupied} - A^3x(k) \\ \vdots \\ (N-1) * P_{BD,occupied} - A^px(k) \end{bmatrix} \tag{4.17}$$

, where $x(k)$ is the initial state condition at the time step QP solver is initialized.

The manipulations outlined above effectively transform our MPC controller problem, seeking the optimal control input over a finite horizon with constraints, into a QP problem. By employing qpOASES as the QP solver, an input series with length as prediction horizon, N, is computed by the solver as the optimal result for current system conditions. Given the receding control strategy inherent to MPC, only the first element of this solved input series is adopted for $\overline{CWND}$. To determine the actual CWND size, a linear shift is applied by adding the constant $\frac{R_{out} * P_{BD,max}}{R_{in,max} * N}$, to $\overline{CWND}$. When a new ACK is received by the sender, it signifies the next time step, $k+1$. Consequently, the QP problem must be reformulated and resolved, again with the first element of the new control series being adopted as the input.

Chapter 5

Experiment Results and Discussion

5.1 Experiment Environment and Setting

5.1.1 Experiment Setup

In this thesis, all experiments are simulated using NS-3 [36], an open-source network simulator written in C++. For solving the QP problems formulated by the MPC controller, we utilize qpOASES [35], a C++ based open-source QP solver. Additionally, for ease of program development, we employ Eigen [55], an open-source template library for linear algebra in C++. All these libraries and modules are seamlessly integrated into the NS-3 simulator environment and its build system, Waf [56].

The implementation of the MPCC requires INT technology to store information at packets' header such as the number of flows and egress queue length for each switch a packet traverses. In Fig. 5.1, the INT overhead format of MPCC-enabled packets is shown, where Q represents the queue length at the egress port of a particular hop, N denotes the number of flows passing through this hop and nhop indicates the number of switch hops traversed by this packet. Each hop's queue length, Q, is stored using 32 bits, while each hop's flow number, N, uses 16 bits. The number of total past switches, nhop, is stored using 16 bits. Consequently, 6 bytes are required to store data for each hop. Assuming the maximum number of switch hops a packet can traverse is five, and with an additional 2 bytes allocated for nhop, the maximum INT overhead for each packet totals 32 bytes.

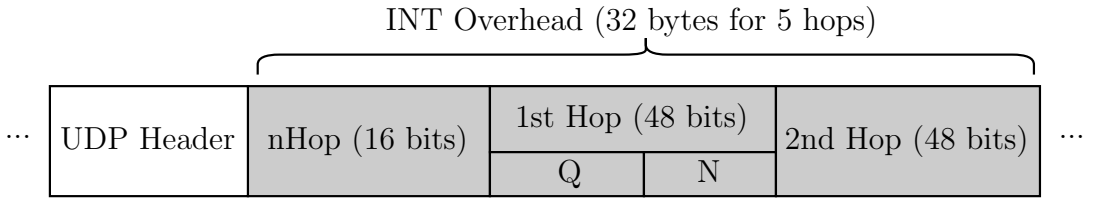


Figure 5.1: INT Overhead Format of MPCC

5.1.2 Baselines, Benchmarks and Test Metrics

For the upcoming experiments, we have selected four widely recognized and deployed CC algorithms as baselines: DCQCN, HPCC, TIMELY, and DCTCP. These experiments are conducted in either dumbbell or fattree topology. In assessing performance, we commonly adopt the following metrics for evaluation:

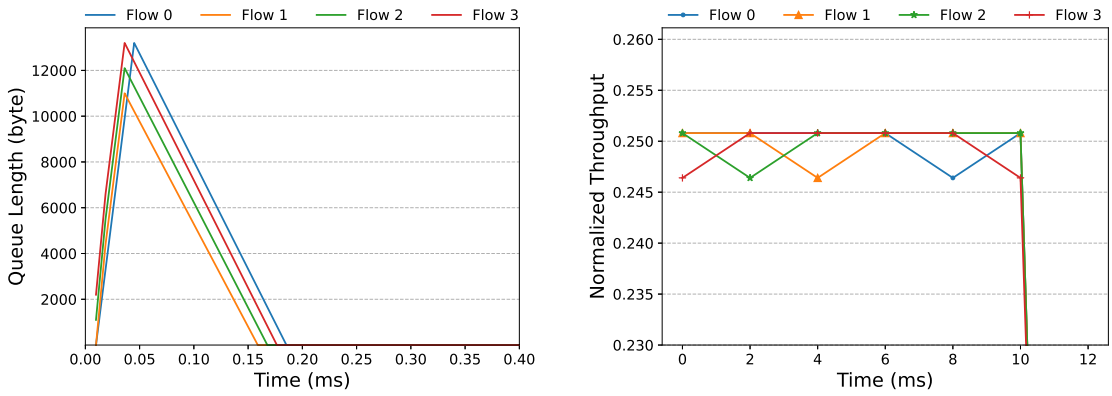
- **FCT slow down**: Flow completion time (FCT) slow down is a dimensionless factor. It is calculated as the ratio of the actual FCT of this flow runs with other flows by the FCT of this flow when it runs alone in the network. A smaller FCT slow down is preferable.

- **Queue length**: Queue length represents the buffer occupancy at the congested switch. A smaller value is preferable.

- **Throughput**: Throughput refers to the rate of successful data packets that are delivered from the sender to its designated receiver. Higher values indicate better performance.

5.2 Experiment 1: Controller Verification

We established a 4-to-1 dumbbell topology as shown in Fig. 3.1, where four senders attempt to transmit packets to the single receiver. Each link has a link rate of 1.0 Gbps and a 1000 ns link delay. Based on the eq. (2.3), the $maxBDP = 30000bytes$. R_{out} can be determined by the conditions outlined in the Tab. 3.1. R_{in} will be the link rate that is 1.0 Gbps.



(a) Queue Length Response of Switch 1 (b) Normalized Throughput Response

Figure 5.2: System Response for MPCC-Enabled 4-to-1 Flow Setting

Initially, as the flows start transmitting packets at the link rate, the queue length at switch 1 begins to accumulate. Upon receiving the first ACK, which includes INT information, the senders detect congestion at switch 1, indicated by a non-zero queue length and the number of active flows passing through it. Subsequently, the MPCC controller adjusts the CWND size to reduce the queue length, thereby eliminating congestion. This response is illustrated in Fig. 5.2a, which shows the rapid convergence of the queue length at n4 to zero. Meanwhile, throughput and fairness among flows are maintained, as demonstrated in Fig. 5.2b. Here, the normalized throughput is a dimensionless factor that represents the bandwidth occupation proportion of each flow. The normalized throughput is measured in a sampling period of 2 ms.

5.2.1 Controller Weight Selection

In eq. (4.5), there are two weighting factor, W_Q and W_R, that requires for tuning. W_Q represents the quadratic cost for queue length, while W_R denotes the quadratic cost for $\overline{CWND}$. Intuitively, a larger W_Q and a smaller W_R make the controller more sensitive to queue length, leading to more dynamic adjustments in CWND to reduce congestion. However, this can cause throughput fluctuations during the transient response. Conversely, a larger W_R and a smaller W_Q make the controller prioritize maintaining the fairshare CWND size, paying less attention to queue length. This

approach yields smoother throughput during transient responses but may increase queue occupancy and packet delay.

Various W_Q-to-W_R weight ratios were tested, revealing a direct correlation between the ratio and the time taken for the queue length at switch 1 to converge to zero. Specifically, convergence times of 140,334 ns, 149,133 ns, and 175,530 ns were observed for W_Q-to-W_R ratios of 1:1, 1:10, and 1:100, respectively. However, ratios smaller than 1:100 or larger than 1:1 did not further alter the convergence time. For a large ratio, the sender will adjust CWND to zero to reduce the cost. Since the CWND size is bounded by the constraint, it cannot become less than zero, so further increasing the ratio will not reduce the convergence time. If the ratio is too small, the controller will not decrease the CWND size even when the queue length is greater than zero. But the CWND cost term is upper bounded by the in-flight byte. Thus, the queue length will not increase to infinite. Interestingly, the normalized throughput of 0.2464 per flow remains constant across all W_Q-to-W_R ratios, as the transient response is too rapid to be captured within a 2ms throughput sampling period. Decreasing the sampling period is also ineffective, as it introduces additional noise.

While an excessively large W_Q-to-W_R ratio does not significantly improve throughput or convergence time, a larger ratio is still theoretically preferred for optimal controller performance.

5.2.2 Prediction Horizon Selection

The prediction horizon is another critical factor in the performance of the MPC controller. A longer prediction horizon enables the controller to better anticipate future events or disturbances, potentially leading to more informed decision-making. However, an excessively long prediction horizon could introduce unnecessary computational complexity without significantly enhancing performance. Because NS-3 is a discrete network simulator, it will not count the CC algorithm running time consumption. But computing complexity is a crucial factor in real network setup as the

CC is a real-time algorithm.

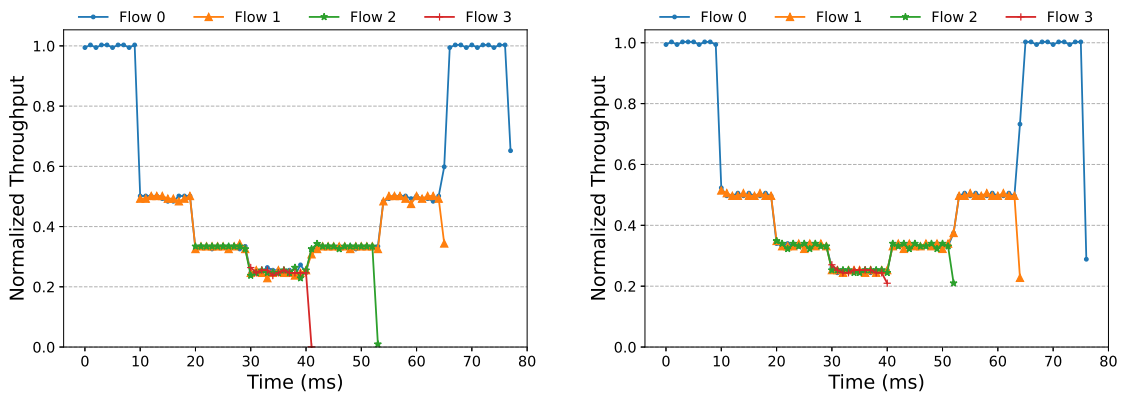
To assess the impact of different prediction horizons, we conducted tests with a fixed W_Q-to-W_R ratio (1:10). The results show that the average time required to solve a single set of QP programming increases with the length of the prediction horizon. Specifically, the times were 255,952 ns, 343,525 ns, 469,736 ns, and 916,831 ns for prediction horizons of 3, 5, 10, and 20, respectively. However, we observed that the time taken for queue length convergence remained consistent across different prediction horizons. Thus, increasing the prediction horizon length did not expedite convergence time but only added to the computational consumption. This phenomenon can be attributed to the fast response of the transient state.

5.3 Experiment 2: Fairness and Convergence

In Section 5.2, we validated the effectiveness of the MPCC controller using a scenario where each of the four senders transmitted one flow simultaneously to a single receiver. The results demonstrated MPCC's ability to effectively manage queue length convergence and maintain throughput fairness among synchronized flows. However, the performance of non-synchronized flows remains to be thoroughly examined.

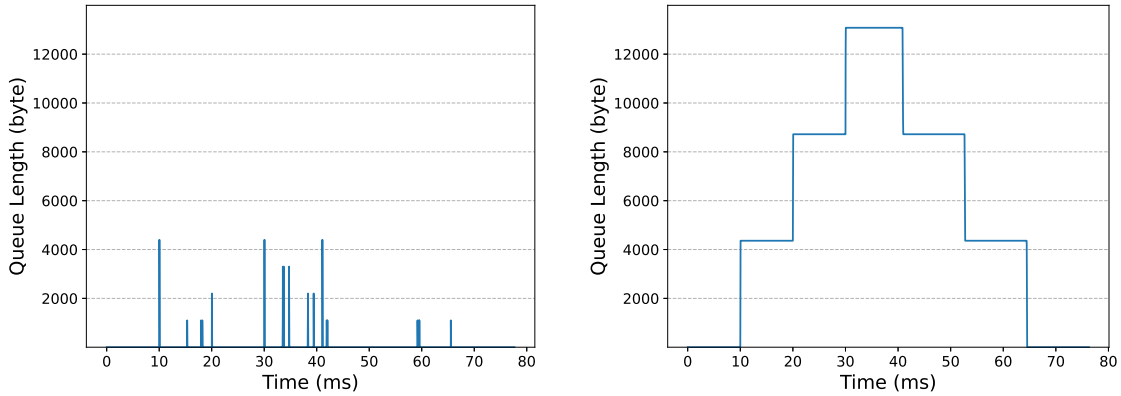
This second experiment maintains the same topology and link settings as in Section 5.2. The only variation is in the flow settings. Specifically, the data lengths for the four flows are set to 4.9 GB, 2.4 GB, 1.4 GB, and 0.3 GB, respectively. Additionally, the start times for these non-synchronized flows are 0.00 sec, 0.01 sec, 0.02 sec, and 0.03 sec, respectively.

The result in Fig. 5.3 illustrates that both MPCC and HPCC successfully converge the throughput to stable and at fairshare rate after a new flow joins the network. Furthermore, when a flow completes, the remaining flows rapidly utilize the newly available bandwidth. However, in Fig. 5.4b, the queue length of switch 1 is always long and occupied. The queue length after the joining of each flow agrees with the upper bound limitation in eq. (4.3), where equals to $(N-1) * Rate_{link} * (2 *$



(a) MPCC (b) HPCC

Figure 5.3: Normalized Throughput Response



(a) MPCC (b) HPCC

Figure 5.4: Queue Length Response of Switch 1

$Delay_{processing} + 2 * Delay_{link}) = (N-1) * 4344bytes$. This observation suggests that the HPCC controller does not adequately respond to changes in flow dynamics. The rapid throughput convergence observed in Fig. 5.3b appears to be a result of passive bandwidth sharing, without active control input or CWND adjustments.

In contrast, MPCC demonstrates an active fast response to the initiation and termination of flows. Fig. 5.4a shows that MPCC effectively manages to reduce the queue length, particularly after the introduction of new flows. The peak queue length occurs shortly after each new flow joins and is quickly reduced through MPCC's CWND adjustments. By utilizing MPCC, the queue length is controlled to converge to zero

while maintaining the fairness and fast convergence of throughput. The observed oscillations in queue length are attributable to the flow counting timeout interval. A shorter interval can increase these oscillations, while a longer interval might slow down the throughput convergence process.

5.4 Experiment 3: Real-World Mixed Traffic

In this experiment, we aim to evaluate the effectiveness of deploying the MPCC in a fattree topology using traffic flows generated from real-world mixed traffic patterns. The traffic flow starting intervals are given by Poisson distribution with λ, where λ represents the average new flow arriving interval and given by:

$$\lambda = ArrivalInterval_{avg} = \frac{1}{BW \times \frac{workload_{\%}}{8 \times flow\_size_{avg}}} \times 10^9,$$

where BW is the bandwidth of the link that is connected to the host. Workload percentage is used to control the new flow arrival interval. *Web_Search* [8] and *FB_Hadoop* [57] are two real traffic flow size distributions are used to generate our flows. Additionally, an 8-to-1 incast traffic that consists of eight flows with 300KB size each will be added to the generated flows to further test the MPCC's robustness.

As shown in Fig. 5.5, the experiment employs a three-level fattree topology consisting of two core switches, four aggregate switches, and four top-of-rack (ToR) switches. Each ToR switch is connected to four hosts, which can function as either receivers or senders. The link connects the core switches to aggregate switches and has a rate of 400 Gbps. The link connects the aggregate switches to ToR switches and has a rate of 400 Gbps. The link rate is 100 Gbps for ToR switches to servers connections. The link delay for all connections is 1000 ns.

In fattree topology, the number of connections to a switch, the link rate and the delay of each connection at each level are the same symmetrically. This means any potential paths chosen by the routing protocol are equivalent in the sense of rate or delay for a packet, provided the number of switch hops (nhop) it traverses is the same.

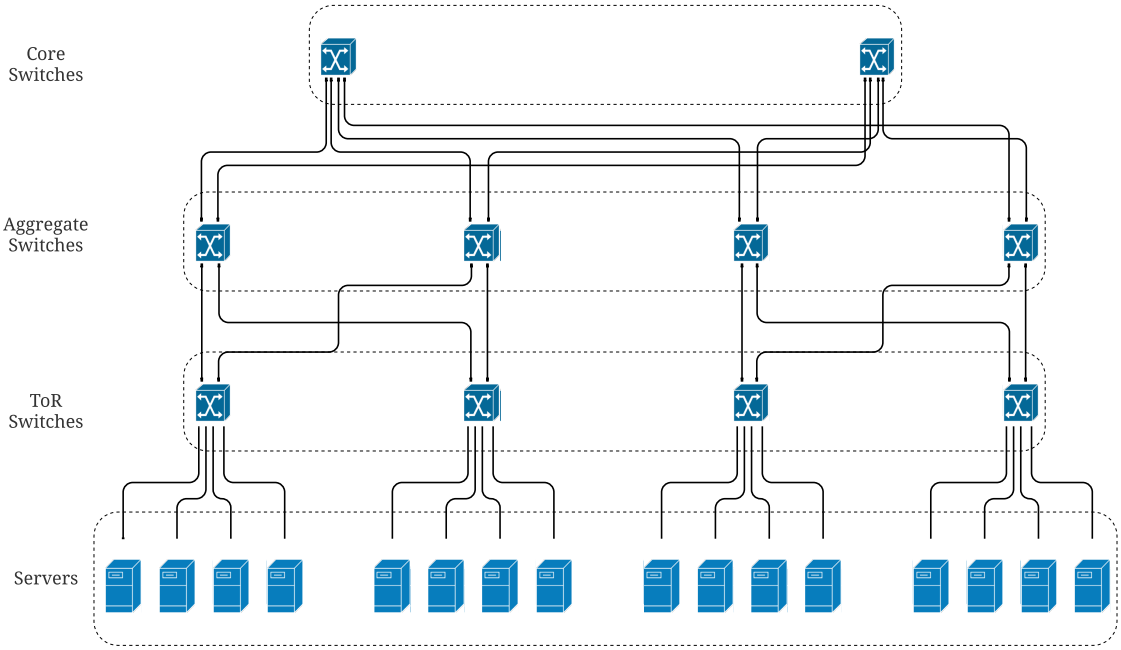


Figure 5.5: A Three-Level Fattree Topology

Any flows transmitted through the same hop experienced the same network conditions except for different queue lengths, which are the same as dumbbell topology. Thus, the state-space equation constructed in Chapter 3 can be extensively applied to fattree topology. Therefore, an additional algorithm is required to perform before determining the $Rate_{out}$ value as shown in Tab. 3.1. This algorithms determines $Rate_{link}$ as detailed in Algorithm 2:

In Fig. 5.7, the FCT slow down across different flow sizes for traffic generated by the *Web_Search* traffic are summarized. In Fig. 5.9, the FCT slow down across different flow sizes for traffic generated by the *FB_Hadoop* traffic is summarized. Lower FCT slow down values are preferable in network performance. Our findings indicate that MPCC always achieves less FCT slow down than TIMELY and DCQCN for both *Web_Search* and *FB_Hadoop* traffic distributions with and without incast.

When comparing MPCC with DCTCP and HPCC, MPCC is capable of keeping the FCT slow down lower for relatively smaller flow sizes than other CCs while maintaining comparable FCT slow down to larger flow sizes for both *Web_Search* and *FB_Hadoop* traffic distributions. This outcome shows that MPCC is more beneficial

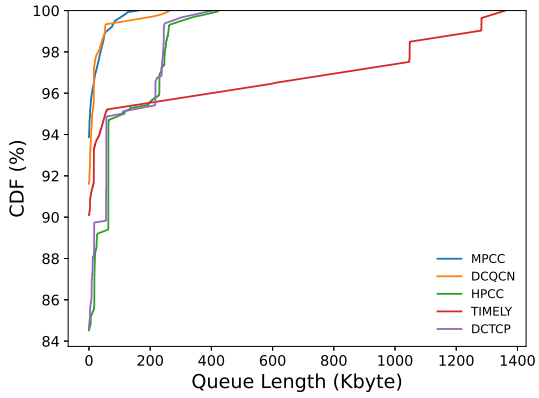
Algorithm 2 *R_{link}Determination*

```
1: if nhop = 1 then
2:    R_link = 100 Gbps
3: else if nhop = 3 then
4:    if Maximum Qlen is at hop 0 or 1 then
5:       R_link = 400 Gbps
6:    else if Maximum Qlen is at hop 2 then
7:       R_link = 100 Gbps
8:    end if
9: else if nhop = 5 then
10:   if Maximum Qlen is at hop 0 or 1 or 2 or 3 then
11:      R_link = 400 Gbps
12:   else if Maximum Qlen is at hop 4 then
13:      R_link = 100 Gbps
14:   end if
15: end if
```

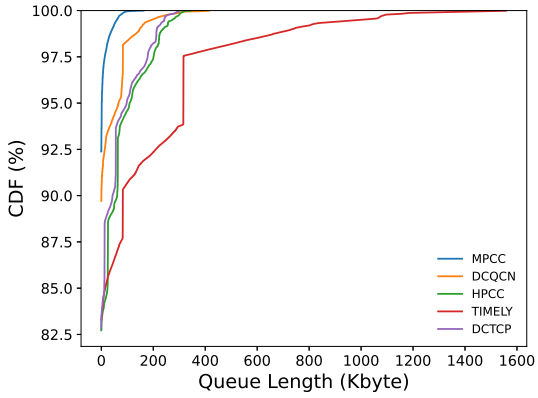
to smaller flows than larger flows, which is attributed to the fast convergence capabilities of MPCC. In scenarios with long-lived large-sized flows existing in the networks, MPCC will fast share their bandwidth with the newly joined flows. Therefore, shorter flows can be completed in less time than other CCs by implementing MPCC. The trade-off is the FCT slow down improvement for larger flow size is limited. Nonetheless, MPCC guarantees the FCT fairness for the larger flows among other CCs while reducing the FCT for smaller flows.

The enhanced FCT performance is achieved by overall lowered queue lengths at switches. In Fig. 5.6 and Fig. 5.8, the running time queue length cumulative distribution function for all switches are shown for both *Web_Search* and *FB_Hadoop* traffic distributions with and without incast. These results demonstrate that MPCC consistently maintains the minimum queue length among the compared CCs. This takes the advantage that using the most congestion switch as the network state for MPC controlling. Then, the less queue length will reduce the RTT for each packet and reduce the FCT. The queue length is proved to be the key parameter to FCT. Furthermore, the less queue length occupation implies less possibility for a packet drop at switches. As more queue lengths in the switch are available, the network will

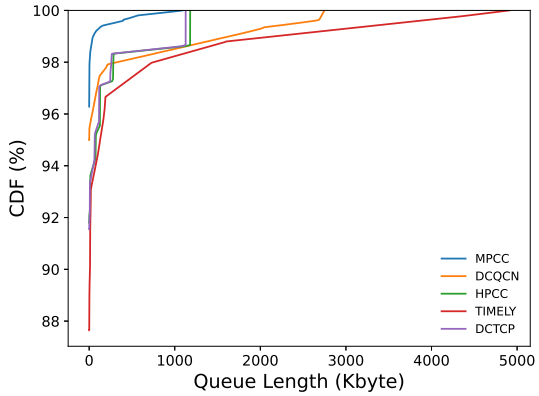
be more resilient against unideal conditions, such as micro bursts, and guarantee the network's smooth operation.



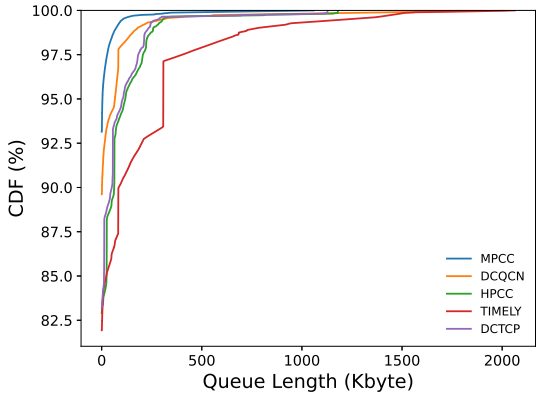
(a) 30% Ave. Load without Incast



(b) 60% Ave. Load without Incast

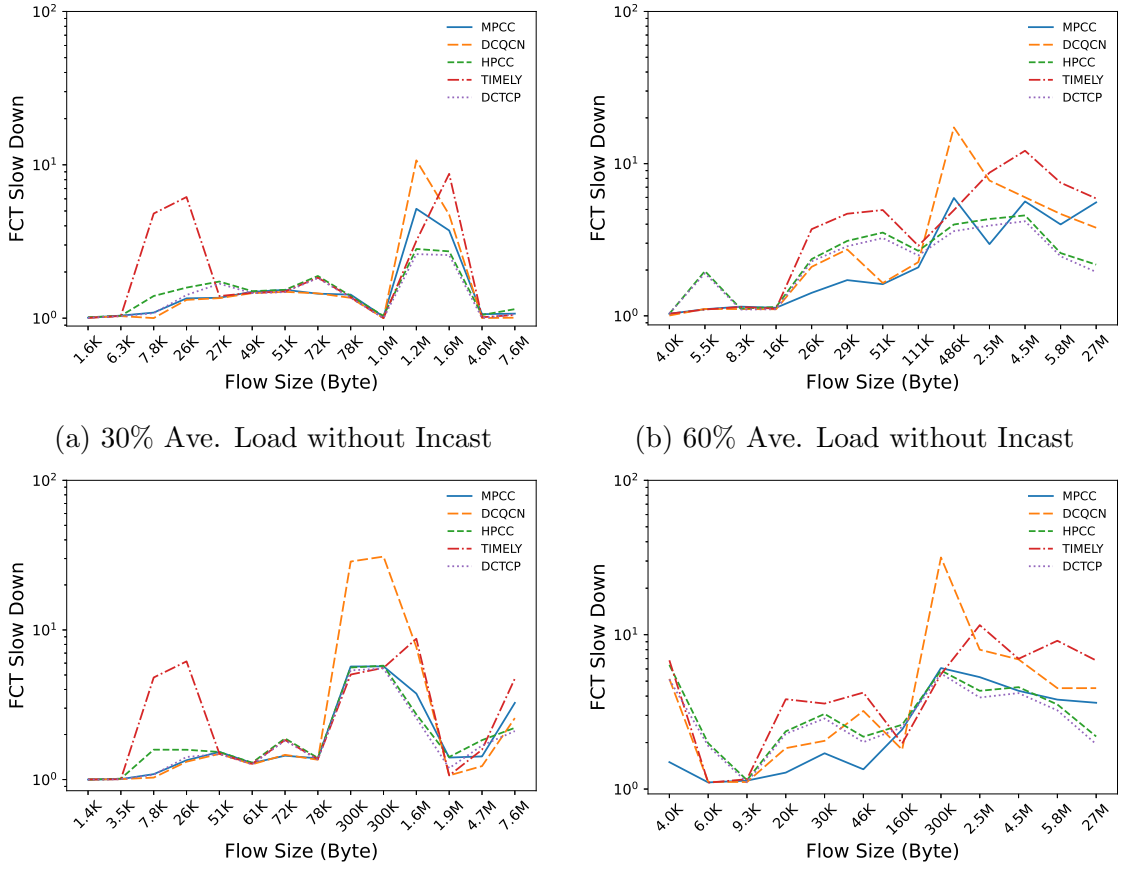


(c) 30% Ave. Load with Incast



(d) 60% Ave. Load with Incast

Figure 5.6: Queue Length CDF with *Web_Search* under Various Load



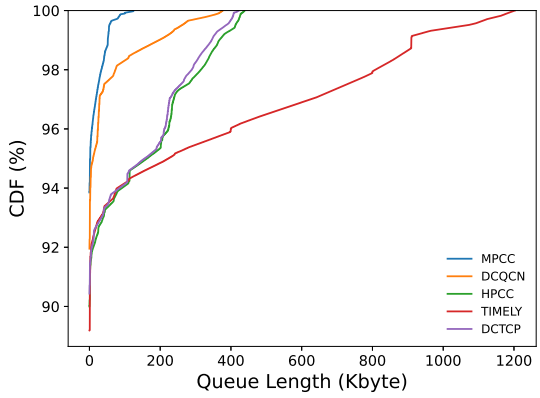
(a) 30% Ave. Load without Incast

(b) 60% Ave. Load without Incast

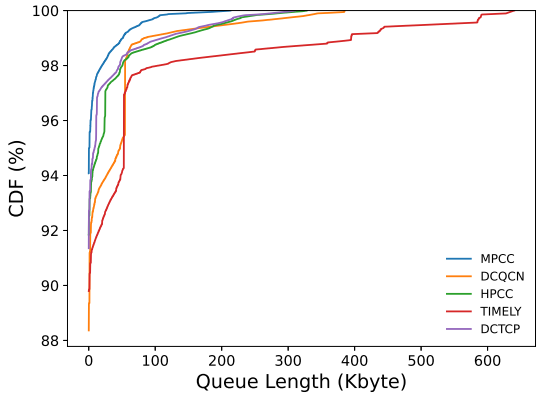
(c) 30% Ave. Load with Incast

(d) 60% Ave. Load with Incast

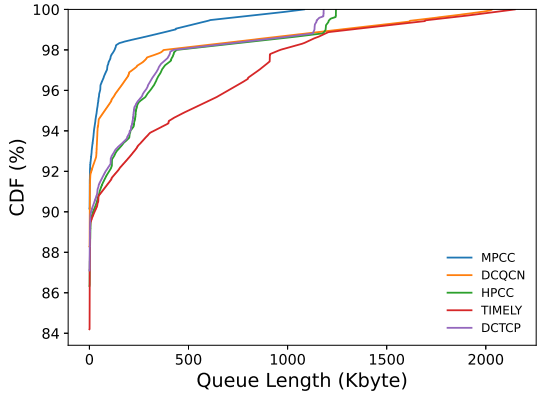
Figure 5.7: FCT Slow Down vs. Flow Size with *Web_Search* under Various Load



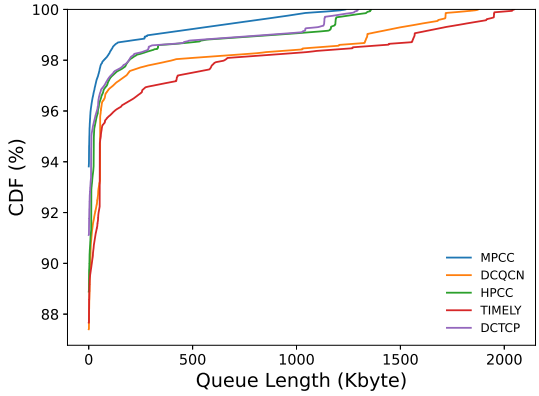
(a) 30% Ave. Load without Incast



(b) 60% Ave. Load without Incast

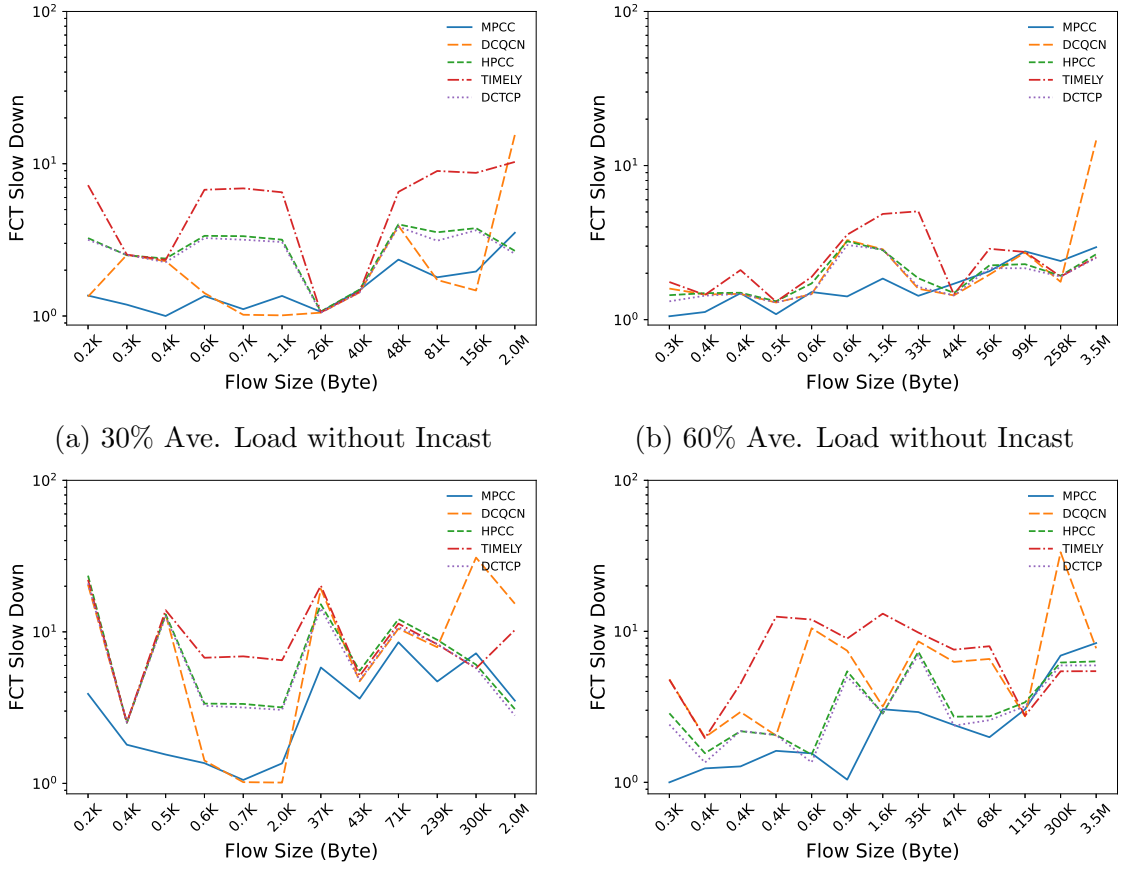


(c) 30% Ave. Load with Incast



(d) 60% Ave. Load with Incast

Figure 5.8: Queue Length CDF with *FB_Hadoop* under Various Load



(a) 30% Ave. Load without Incast

(b) 60% Ave. Load without Incast

(c) 30% Ave. Load with Incast

(d) 60% Ave. Load with Incast

Figure 5.9: FCT Slow Down vs. Flow Size with *FB_Hadoop* under Various Load

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we focus on developing an enhanced CC algorithm tailored for RDMA-enabled DCNs. Our analysis identifies the egress queue length of intermediate switches as a critical factor influencing packet delay and the likelihood of packet drops. Utilizing INT technology alongside detailed DCN topology information, we have derived a delayed mathematical discrete state-space model by the first principle. This model, based on the continuity equation, describes the dynamic relationship between the queue length and the size of the CWND.

Building on this discrete state-space equation, we have developed the MPCC algorithm. It stands out by addressing shortcomings observed in other CC algorithms, such as the lack of mathematical proof and slow convergence. MPCC is a linearly constrained MPC controller that embraces the state-space equation for optimal control, integrating network condition constraints directly into the model.

Experimental results demonstrate that the MPCC-enabled networks can effectively regulate queue lengths, reducing them to nearly zero, while ensuring throughput and fairness among different flows. MPCC exhibits proficiency in bandwidth sharing and achieving rapid convergence for newly joined flows. It also demonstrates superior performance in real-world mixed traffic scenarios. For instance, FCT slow down for smaller-sized flows is significantly reduced under MPCC, while it maintains compa-

rable performance for larger-sized flows relative to other CC algorithms.

6.2 Future Work

Implementing the MPCC algorithm in real DCN environments poses unique challenges, particularly due to computational time constraints. Unlike in NS-3 simulations, where simulation time does not increase until the CC algorithm code execution is complete, the computation time for QP in real DCNs is critical, as it can potentially degrade network performance. To address this, two potential solutions are proposed:

1) *Select Proper Prediction Horizon*: As detailed in Subsection 5.2.2, extending the prediction horizon length does not necessarily enhance the convergence rate, given the rapid control reactions of the system. Therefore, opting for a shorter prediction horizon can effectively reduce the computation time required for QP programming. This consideration leads to our second solution.

2) *Optimize MPCC Algorithm*: Since the prediction horizon length is not directly correlated to the convergence rate due to its rapid nature of control actions, reducing the frequency of control actions could be beneficial. For instance, instead of modifying the CWND size in response to every ACK, adjusting the CWND less frequently (e.g., after receiving several ACKs) might be more effective. This approach may allow the prediction horizon to regain its significance. Moreover, the idle time while the sender awaits multiple ACKs can be utilized for computing the QP programming, thus mitigating additional computational delays.

These strategies aim to enhance the practical applicability of MPCC in real-world DCNs by optimizing computational efficiency without compromising the algorithm's performance.

Bibliography

[1] H. Zhang, Y. Zhao, C. Liu, and J. Xu, "Commercial open line system optimization for high speed data center optical transmission," in *2021 IEEE 4th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, 2021, pp. 312–315.

[2] Y. Zhu *et al.*, "Congestion control for large-scale rdma deployments," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, 523–536, Aug. 2015, ISSN: 0146-4833.

[3] C. Guo *et al.*, "Rdma over commodity ethernet at scale," *SIGCOMM*, SIGCOMM '16, 202–215, 2016.

[4] Y. Gao *et al.*, "When cloud storage meets RDMA," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, USENIX Association, Apr. 2021, pp. 519–533, ISBN: 978-1-939133-21-2. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/gao.

[5] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN systems*, vol. 17, no. 1, pp. 1–14, 1989.

[6] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, 64–74, Jul. 2008, ISSN: 0163-5980.

[7] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "Fast tcp: Motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, 2006.

[8] M. Alizadeh *et al.*, "Data center tcp (dctcp)," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, 63–74, Aug. 2010, ISSN: 0146-4833.

[9] R. Mittal *et al.*, "Timely: Rtt-based congestion control for the datacenter," *SIGCOMM*, vol. 45, no. 4, 537–550, Aug. 2015, ISSN: 0146-4833.

[10] Y. Li *et al.*, "Hpcc: High precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19, Beijing, China: Association for Computing Machinery, 2019, 44–58, ISBN: 9781450359566.

[11] J. Zhang *et al.*, "Rcc: Enabling receiver-driven rdma congestion control with congestion divide-and-conquer in datacenter networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 1, pp. 103–117, 2023.

[12] V. Jacobson, "Congestion avoidance and control," in *Symposium Proceedings on Communications Architectures and Protocols*, ser. SIGCOMM '88, Stanford, California, USA: Association for Computing Machinery, 1988, 314–329, ISBN: 0897912799.

[13] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '02, Pittsburgh, Pennsylvania, USA, 2002, 89–102, ISBN: 158113570X.

[14] N. Dukkipati, N. McKeown, and A. G. Fraser, "Rcp-ac: Congestion control to make flows complete quickly in any environment," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 2006, pp. 1–5.

[15] S. Athuraliya, S. Low, V. Li, and Q. Yin, "Rem: Active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, 2001.

[16] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '00, Stockholm, Sweden: Association for Computing Machinery, 2000, 151–160, ISBN: 1581132239.

[17] F. Yanfie, R. Fengyuan, and L. Chuang, "Design a pid controller for active queue management," in *Proceedings of the Eighth IEEE Symposium on Computers and Communications. ISCC 2003*, 2003, 985–990 vol.2.

[18] P. Wang, H. Chen, X. Yang, and Y. Ma, "Design and analysis of a model predictive controller for active queue management," *ISA Transactions*, vol. 51, no. 1, pp. 120–131, 2012, ISSN: 0019-0578.

[19] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," vol. 24, no. 4, 24–35, Oct. 1994, ISSN: 0146-4833.

[20] C. Lu, Z. Tang, W. Peng, G. Lv, and P. Xun, "Running p4 programs on general programmable network interconnection chips," in *2023 Fourth International Conference on Frontiers of Computers and Communication Engineering (FCCE)*, 2023, pp. 1–6.

[21] L. Tan *et al.*, "In-band network telemetry: A survey," *Computer Networks*, vol. 186, p. 107 763, 2021.

[22] *Ieee. 802.11qbb. priority based flow control*, 2011. [Online]. Available: https://standards.ieee.org/ieee/802.1Qbb/4361/.

[23] J. Shao, X. Li, M. Li, S. Liu, and Y. Xu, "S-pfc: Enabling semi-lossless rdma network with selective response to pfc," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, 2023, pp. 1135–1141.

[24] G. Chen *et al.*, "Fuso: Fast multi-path loss recovery for data center networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1376–1389, 2018.

[25] C. Ekaputri and A. Syaichu-Rohman, "Model predictive control (mpc) design and implementation using algorithm-3 on board spartan 6 fpga sp605 evaluation kit," in *2013 3rd International Conference on Instrumentation Control and Automation (ICA)*, 2013, pp. 115–120.

[26] Z. Ma, S. Saeidi, and R. Kennel, "Fpga implementation of model predictive control with constant switching frequency for pmsm drives," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2055–2063, 2014.

[27] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002, ISSN: 0005-1098.

[28] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control*, vol. 18, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:16049363.

[29] J. L. Jerez *et al.*, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, 2014.

[30] H. Efheij, A. Albagul, and N. Ammar Albraiki, "Comparison of model predictive control and pid controller in real time process control system," in *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2019, pp. 64–69.

[31] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1094–1099.

[32] S. Kouro, P. Cortes, R. Vargas, U. Ammann, and J. Rodriguez, "Model predictive control—a simple and powerful method to control power converters," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1826–1838, 2009.

[33] R. Bergman, L. Phillips, and C Cobelli, "Physiologic evaluation of factors controlling glucose tolerance in man: Measurement of insulin sensitivity and beta-cell glucose sensitivity from the response to intravenous glucose," *Journal of Clinical Investigation*, vol. 68, pp. 1456–146, 1981.

[34] S. Lynch and B. Bequette, "Estimation-based model predictive control of blood glucose in type i diabetics: A simulation study," in *Proceedings of the IEEE 27th Annual Northeast Bioengineering Conference (Cat. No.01CH37201)*, 2001, pp. 79–80.

[35] *Qpoases*, Accessed: 2023-10-04, 2023. [Online]. Available: https://github.com/coin-or/qpOASES.

[36] *Network simulator 3*, Accessed: 2023-10-04, 2023. [Online]. Available: https://www.nsnam.org/.

[37] P. Wysocki *et al.*, "Broad-band erbium-doped fiber amplifier flattened beyond 40 nm using long-period grating filter," *IEEE Photonics Technology Letters*, vol. 9, no. 10, pp. 1343–1345, 1997.

[38] T. Horiguchi and M. Tateda, "Botda-nondestructive measurement of single-mode optical fiber attenuation characteristics using brillouin interaction: Theory," *Journal of Lightwave Technology*, vol. 7, no. 8, pp. 1170–1176, 1989.

[39] J. Armstrong, "Ofdm for optical communications," *Journal of Lightwave Technology*, vol. 27, no. 3, pp. 189–204, 2009.

[40] P. Poggiolini, "The gn model of non-linear propagation in uncompensated coherent optical systems," *Journal of Lightwave Technology*, vol. 30, no. 24, pp. 3857–3879, 2012.

[41] *Transmission Control Protocol*, RFC 793, Sep. 1981. DOI: 10.17487/RFC0793. [Online]. Available: https://www.rfc-editor.org/info/rfc793.

[42] D. Yang *et al.*, "An optical fiber comprehensive analysis system for spectral-attenuation and geometry parameters measurement," in *2017 Conference on Lasers and Electro-Optics Pacific Rim (CLEO-PR)*, 2017, pp. 1–2.

[43] D. Yang *et al.*, "An optical fiber comprehensive analysis system for spectral-attenuation and geometry parameters measurement," in *2017 Conference on Lasers and Electro-Optics Pacific Rim (CLEO-PR)*, 2017, pp. 1–2.

[44] I. Rhee *et al.*, *CUBIC for Fast Long-Distance Networks*, RFC 8312, Feb. 2018. DOI: 10.17487/RFC8312. [Online]. Available: https://www.rfc-editor.org/info/rfc8312.

[45] Deland-Han *et al.*, *Overview of tcp/ip performance*, Accessed: 2023-10-12, 2023. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/overview-of-tcpip-performance.

[46] M. H. Eiselt and F. Azendorf, "Accurate measurement of propagation delay in a multi-span optical link," in *2019 International Topical Meeting on Microwave Photonics (MWP)*, 2019, pp. 1–3.

[47] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.*, vol. 3, 2004, 1629–1634 Vol.3.

[48] Cisco, *Cisco data center networking blueprint for ai/ml applications*, 2023. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/cisco-data-center-networking-blueprint-for-ai-ml-applications.html.

[49] W. Bai *et al.*, "Empowering azure storage with rdma," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, USENIX, Apr. 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/empowering-azure-storage-with-rdma/.

[50] *Rdma and roce for ethernet network efficiency performance*, Accessed: 2023-10-06, 2014. [Online]. Available: https://support.mellanox.com/s/productdetails/a2v50000000XcRAAA0/roce.

[51] *Infiniband*, Accessed: 2023-10-06, 2023. [Online]. Available: https://www.infinibandta.org/.

[52] J. H. Lienhard IV and J. H. Lienhard V, *A Heat Transfer Textbook*, 5th. Cambridge, MA: Phlogiston Press, Aug. 2020, pp. 6–7, Version 5.10. [Online]. Available: http://ahtt.mit.edu.

[53] A. Singh *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15, London, United Kingdom: Association for Computing Machinery, 2015, 183–197, ISBN: 9781450335423.

[54] Cisco, *Cisco catalyst 9300 series switches data sheet*, 2023. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9300-series-switches/nb-06-cat9300-ser-data-sheet-cte-en.html.

[55] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, 2010. [Online]. Available: http://eigen.tuxfamily.org.

[56] T. Nagy *et al.*, *Waf, the build system*, 2005. [Online]. Available: https://waf.io/.

[57] A. Roy *et al.*, "Inside the social network's (datacenter) network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15, London, United Kingdom: Association for Computing Machinery, 2015, 123–137, ISBN: 9781450335423.

Appendix A: First Appendix

A.1 Flow counting for Egress Port

This part of the appendix outlines the method for counting active flow numbers in a specific egress queue of network switches. For each egress port, the switch initializes a set of timers and a flow counter, which are instrumental in tracking active flows.

- Packet Arrival: When a packet enters a designated egress queue, the switch extracts four key pieces of information from the packet header: SIP, DIP, SPORT, and DPORT. Each flow is distinguishable by at least one unique attribute in these four parameters. Even for flows that have the same SIP and DIP, their DPORT will still be different.

- Unique Flow Identification: A hash function is applied to these four parameters, generating a unique key for each individual flow. The switch then initiates a timer for each unique key within a specific egress queue.

- Flow Counter and Timer Management: Upon receiving a packet with a new (unseen) flow hash key, the switch increments its flow counter and sets up a new timer for that flow. Each egress queue, therefore, maintains a set of timers and a corresponding flow counter. If a packet arrives with a previously seen hash key within a certain time interval, the associated timer is reset, and the value of the flow counter remains. In contrast, if no packet for an existing flow is received within the designated interval, the corresponding timer is deleted, and the flow counter is decremented.

- Header Information Update: As a packet leaves the egress queue, the current flow number, along with the egress queue length at that instant, is pushed to the packet's INT header. This process is replicated across every switch the packet traverses.

A.2 Discrete Interval Estimation

The discrete interval in network communications is defined as the time gap between two successive ACK signals. This interval is a crucial parameter for discretizing the eq. (3.3). RTT for a given flow path can be estimated using the eq. (2.2). Utilizing this RTT and the size of the CWND (in-flight bytes), we can estimate the average interval for the current batch of in-flight bytes using the following formula:

$$\Delta t = \frac{1080 * D}{W}, \tag{A.1}$$

where:

- 1080 is the number of bytes of one packet (1000 bytes for payload, 48 bytes for protocol header, 32 bytes for INT information).
- D is the RTT of the current flow path.
- W is current CWND in byte.