TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

LIST OF ABREVIATIONS

5G 5th Generation Mobile Networks. IX, 2, 17, 94, 97, 107

ATM Asynchronous Transfer Mode. 12

BBU baseband unit. 18, 94, 96, 97, 104

BDP Bandwidth Delay Product. 14

BER Bit Error Rate. 20

C-RAN Centralized Radio Access Network. 18, 94, 97

CAPEX CAPital EXpenses. 2, 7, 19, 93, 96

CBFC Credit-based Flow Control. 12

CEE Converged Enhanced Ethernet. 2, 7

CNM Congestion Notification Message. 11, 64–66, 68, 69

CP Congestion Point. 10, 11, 64

CPRI Common Public Radio Interface. IX, 2, 7, 19, 20, 93, 95–97, 99–104, 106, 107

CPU Central processing unit. 2, 5, 60, 61

DCB Data Center Bridging. 1, 2, 7, 8, 59, 60, 62, 107

DCN Data Center Network. 1, 5, 7, 59, 88

DCQCN Data Center QCN. 16, 90

DCTCP Data Center TCP. 15, 16, 89, 90

DPDK Data Plane Development Kit. 61, 62, 86

DTSCoE Distributed Timeslot Scheduler for CPRI over Ethernet. 93, 99, 100, 102–106

ECMP Equal-cost multi-path routing. 9, 62

ETN Ethernet Transport Network. 96

FCoE Fibre Channel over Ethernet. 1, 7, 9, 59

FCP Fastpass Control Protocol. 13

FECN Forward Explicit Congestion Notification. 14

HetFlow Heterogeneous Flow. 59, 61, 62, 65–71, 73–78, 80, 81, 83–91, 107

HOL Head Of Line. IX, 3, 8–10, 20, 62, 63

HPC High Performance Computing. 1, 59

HULL High-bandwidth Ultra-Low Latency. 15

I/Q In-phase and Quadrature-phase. 18, 94, 98

IB InfiniBand. 2, 61

iSCSI Internet Small Computer System Interface. 7, 9

NoP-ECCP No-Probe ECCP. 20

NTCF Non-Time-Critical Frame. 98

NTP Network Time Protocol. 98

OAM Operations, Administration and Maintenance. IX, 2, 7, 20, 93, 96

OPEX OPerating EXpenses. 2, 7, 19, 93, 96

PFC Priority-based Flow Control. 2, 8–10, 12, 60–64, 106

ppb parts-per-billion. 98

PQ Phantom Queue. 16

PTP Precision Time Protocol. 98

QCN Quantized Congestion Notification. 2, 8, 10, 12, 16, 21, 60, 61, 64, 68, 76–78, 83–86, 88, 90, 107

RDMA Remote Direct Memory Access. 1, 2, 61

RE radio equipment. 2, 7, 18, 19, 93–96, 98, 103

REC Radio Equipment Control. 2, 7, 18, 19, 93–96, 98, 103

RoCE RDMA Over Converged Ethernet. 2, 61, 88

RoE Radio over Ethernet. 3, 18

RP Reaction Point. 10, 64

RRH Remote Radio Head. 18, 19, 94, 95, 97

RTT Round-trip Time. IX, 2, 3, 6, 12, 14, 16, 59–62, 64, 70, 84–86, 90

SAN Storage Area Network. 1, 59

SDN Software-Defined Networking. 7

SRP Stream Reservation Protocol. 98, 99

Sync-E Synchronous Ethernet. 98, 102

TCF Time-Critical Frame. 98

TCP Transmission Control Protocol. 5–7, 14, 15, 59, 60, 69, 84, 89

ToD Time of Day. 96

TSN Time-Sensitive Network. 98, 99

VC Virtual Channel. 12

VLAN Virtual Local Area Network. 1

VM Virtual Machine. 1

VXLAN Virtual eXtensible Local Area Network. 1, 59, 88

INTRODUCTION

Ethernet has experienced huge capacity-driven growth recently from 10 Gbps up to 100 Gbps. The advantages of Ethernet are threefold 1) the low cost of equipment, 2) the scalability, as well as 3) the ease of operations, administration and maintenance (OAM). These features make Ethernet the best candidate to provide transport network for many application e.g. Data Center Network (DCN), converged Storage Area Network (SAN), High Performance Computing (HPC), cloud computing and Fibre Channel over Ethernet (FCoE). In this research, we explore the possibility of achieving a lossless, or more precisely, drop-free Ethernet. Further, we study the effect of this lossless Ethernet on several applications, namely i) switch fabric in routers, ii) data center network, iii) Remote Direct Memory Access (RDMA), iv) Common Public Radio Interface (CPRI) over Ethernet.

Switch fabric in routers requires very tight characteristics in term of packet loss, fairness in bandwidth allocation, low latency and no head-of-line blocking. Such attributes are traditionally resolved using specialized and expensive switch devices. With the enhancements that are presented by IEEE Data Center Bridging (DCB) (802.1, 2013) for Ethernet network, we explore the possibility of using commodity Ethernet switches to achieve scalable, flexible, and more cost-efficient switch fabric solution, while still guaranteeing router characteristics.

In addition, the rise of DCN facilitates new applications such as SAN and Virtual Machine (VM) automated deployment and migration that require high data rate, ultra-low latency and packet loss. Additionally, DCN is required to support layer-two applications such as Virtual Local Area Network (VLAN) and Virtual eXtensible Local Area Network (VXLAN) that provide flexible workload placement and layer 2 segmentation. Because Ethernet is the most widely used transport network in data center fabric, we study the possibility of achieving a lossless transport layer to support these applications.

Due to Ethernet widespread, other technologies are migrating to Ethernet such as RDMA. RDMA technology offers high throughput, low latency, and low Central processing unit (CPU) overhead, by allowing network interface cards (NICs) to transfer data in and out of host's memory directly. Originally, RDMA requires InfiniBand (IB) network protocol/infrastructure to operate. RDMA over IB requires adopting new network infrastructure which has experienced limited success in the enterprise data centers. RDMA Over Converged Ethernet (RoCE) v1 (Association *et al.*, 2010) and v2 (Association *et al.*, 2014) are presented as new network protocols which permit performing RDMA over Ethernet network. RoCE presents an intermediate layer with IB as an upper interface to support RDMA and Ethernet as a lower interface. This allows using RDMA over standard Ethernet infrastructure with specific NICs that support RoCE. Such application requires a robust and reliable Ethernet network which raises the need for an Ethernet congestion control protocol.

Finally, we investigate providing a transport network for CPRI traffic in the 5G network fronthaul. By encapsulating CPRI traffic over Ethernet, significant savings in CAPital EXpenses (CAPEX) and OPerating EXpenses (OPEX) can be achieved. In addition, the OAM capabilities of Ethernet provide standard methods for network management and performance monitoring. Thus, Ethernet is proposed for the 5G fronthaul to transport the CPRI traffic between the radio equipment (RE) and theRadio Equipment Control (REC). In this research, we investigate providing lossless Ethernet using the enhancements that are provided in the DCB standards within IEEE 802.1 task group. DCB standards are also known as Converged Enhanced Ethernet (CEE) and it comprises Priority-based Flow Control (PFC) (IEEE Standard Association, 2011) and Quantized Congestion Notification (QCN) (IEEE 802.1Qau, 2010; Alizadeh *et al.*, 2008) that address congestion control in Ethernet layer.

In this context, we aim to design an Ethernet congestion control mechanism that achieves i) high link utilization, ii) close-to-zero (ultra low) queue length, iii) low latency, iv) fairness

between flows of different packet sizes, and v) fairness between flows of different RTTs. These mechanisms are to be implemented in Ethernet layer; hence, It should consider these Ethernet limitations:

- No per-packet ACK in the Ethernet network.
- The traffic is highly bursty in the Ethernet network.
- The switch buffer size is much smaller, comparing to router buffer size.
- Ethernet supports high bandwidth traffics (10Gbps and 100Gbps). Thus, it requires fast convergence algorithm within a minimal delay.
- Round trip time is in the range of microseconds.

The rest of the thesis is organized as follow. Chapter 1 states the research problem and illustrate research work that is related congestion control. In addition, it lists summary of publication that has been done through the course of this research project. Through Chapter 2,3 and 4, we investigate the possibility of achieving lossless/dropless Ethernet network to be used for Several applications. E.g. In Chapter 2, we investigate achieving lossless router fabric using Ethernet network. Furthermore, Chapter 3 investigates the capability of Ethernet network to fulfill data centers requirements including no packet drop, no HOL blocking and fairness between flows of different characteristics (different packet sizes and RTTs). Moreover, we address the potential of using Ethernet network as a transport layer for Radio over Ethernet (RoE) in Chapter 4. In this chapter, we introduced a scheduling algorithm for IEEE 802.1Qbv standard to support the transmission of time-critical flows such as RoE. Our testbed experiments and simulations show that Ethernet has the potential of providing lossless transport network for the aforementioned applications.

CHAPTER 1

RESEARCH PROBLEM

Packet loss has a significant impact on transport network performance. T.V. Lakshman et al. shown in (Lakshman & Madhow, 1997b) that Transmission Control Protocol (TCP) Reno causes link utilization to drop dramatically to 37.9% when a packet loss probability of 10^{-3} is applied to the network. (Andrew S. Tanenbaum, 2011) states that using congestion control increase the network goodput as shown in figure 1.1. The figure depicts that the goodput of lossy networks increases linearly as the network load increases till it starts experiencing packet loss or congestion. Subsequently, the goodput decreases dramatically. On the other hand, the goodput of lossless network increases linearly with network load till the maximum.

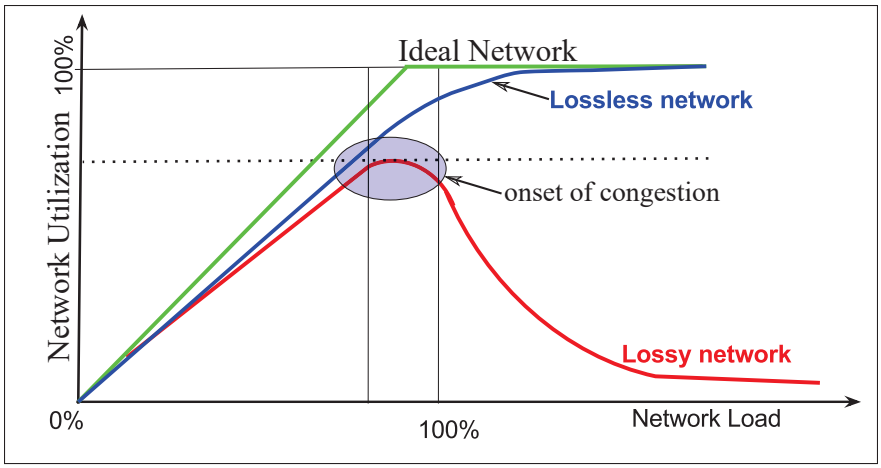


Figure 1.1 Goodput of Lossy and Lossless Networks (Andrew S. Tanenbaum, 2011)

Traditionally, TCP is considered as the main transport protocol on the Internet that is used as well in DCN. TCP has major problems; e.g., TCP reacts on packet loss events whereas packet loss causes huge degradation in the performance of most data center applications. Additionally, (Zhu *et al.*, 2015) reported that TCP consumes, on average, over 20% of CPU power. It also stated that at small packet sizes, CPU becomes the bottleneck and cannot saturate the link because of TCP overhead.

Moreover, Fairness between flows of different packet sizes and different RTTs represents a major challenge to current congestion control protocols. Most network devices detect congestion when their queue level reaches a maximum length in bits, while congestion control mechanisms react per packet. Thus, flows with small packet sizes experience a high number of packet loss than flows with large packet sizes which lead to over-controlling flows with small packet sizes. Therefore, congestion control mechanisms designed for fixed packet size flows cause unfairness and link under-utilization when packer sizes vary (Shah *et al.*, 2012; Wilson, 2008). Further, different RTTs strongly affect the performance of congestion control mechanisms. An experiment is conducted in (Holman *et al.*, 2012) using FreeBSD TCP-NewReno demonstrating that flows with high latency suffer the most when sharing a bottleneck link with low latency flows. Small RTT flows complete more round trips in the same period comparing to large RTT flows which lead to faster recovery. Therefore, small-RTT flows get a higher share of the available bandwidth.

Therefore, in this research we aim to design a lightweight Ethernet congestion control protocol that achieves i) high link utilization, ii) close-to-zero queue length, iii) low latency, iv) fairness between flows of different packet sizes and different RTTs v) with commodity Ethernet switches.

1.1 Motivation & Impact

Switch fabric in routers requires very tight characteristics in term of packet loss, fairness in bandwidth allocation, no head-of-line blocking and low latency. Such attributes are traditionally resolved using specialized and expensive switch devices. In addition, Data center applications require strict characteristics regarding packet loss, fairness, head-of-line blocking, latency, and low processing overhead. Motivated by the emergence of IEEE Data Center Bridging, we explore the possibility of using commodity Ethernet switches to achieve scalable, flexible, and more cost-efficient transport network, while still guaranteeing the aforementioned required characteristics.

On the other hand, the exponential growth in mobile network users and the enormous bandwidth required by new mobile applications raise the need for robust, reliable and cost-efficient transport network. CPRI is currently the most widely used protocol for fronthaul transport between the REC and the RE. However, CPRI has very stringent requirements regarding delay and jitter. Traditionally, these requirements are met using point-to-point fiber optics which increases both CAPEX and OPEX of mobile networks. Besides, using Ethernet as a transport network for fronthaul draws significant attention of both academia and industry. The Ethernet-based fronthaul network provides several advantages such as i) low-cost equipment, ii) sharing existing infrastructure, as well as iii) the ease of OAM. In this research we study the possibility of providing a robust transport layer for fronthaul network to support CPRI over Ethernet.

1.1.1 State of the Art

The raising of DCN (Bilal *et al.*, 2013) and Software-Defined Networking (SDN) (Committee *et al.*, 2012) requires high quality, reliable and stable network particularly in case of congestion. Many DCN applications are very sensitive to packet loss such as FCoE (Croft *et al.*, 2003; Desai *et al.*, 2007) and Internet Small Computer System Interface (iSCSI) (Satran & Meth, 2004). Therefore, many congestion control protocols are presented in the literature to address these requirements. In the following sections, we discuss few research articles that are close to our research subject.

1.1.1.1 Ethernet layer congestion control protocols

Because of the widespread of Ethernet, it has become the primary network protocol that is considered to support both DCN and SDN. Ethernet was originally designed as a best-effort communication protocol, and it does not guarantee frame delivery. Many providers believe that TCP can perform well in case of network congestion. However, TCP sender detects congestion and reacts by reducing its transmission rate when segment loss occurs. To avoid this conservative TCP reaction on segments loss, one should minimize packet dropping at layer 2. In this context, IEEE has defined a group of technologies to enhance Ethernet into a lossless

fabric named DCB (802.1, 2013) which are also known as CEE. These technologies aim to create a robust and reliable bridge between data center components through Ethernet network. DCB comprises Ethernet PAUSE IEEE 802.3x, (PFC - 802.1Qbb) (IEEE Standard Association, 2011) and QCN (802.1Qau) (IEEE 802.1Qau, 2010; Alizadeh *et al.*, 2008).

These technologies can be classified based on the reaction point into two categories i) Hop-by-Hop or ii) End-to-End. In hop-by-hop flow control mechanisms, control messages are forwarded from node to node in a store-and-forward manner. Hop-by-hop transport involves the source, destination node, and some or all of the intermediate nodes. Hop-by-hop mechanisms react faster than End-to-End ones. However, it propagates the congestion starting from the congested point backward to the source causing what is known in the literature as congestion spreading or tree saturation effect (Hanawa *et al.*, 1996) (Figure 1.2a). Consequently, it causes HOL blocking. In addition, hop-by-hop mechanisms face scalability issue because it needs to keep per-flow state information at intermediate nodes.

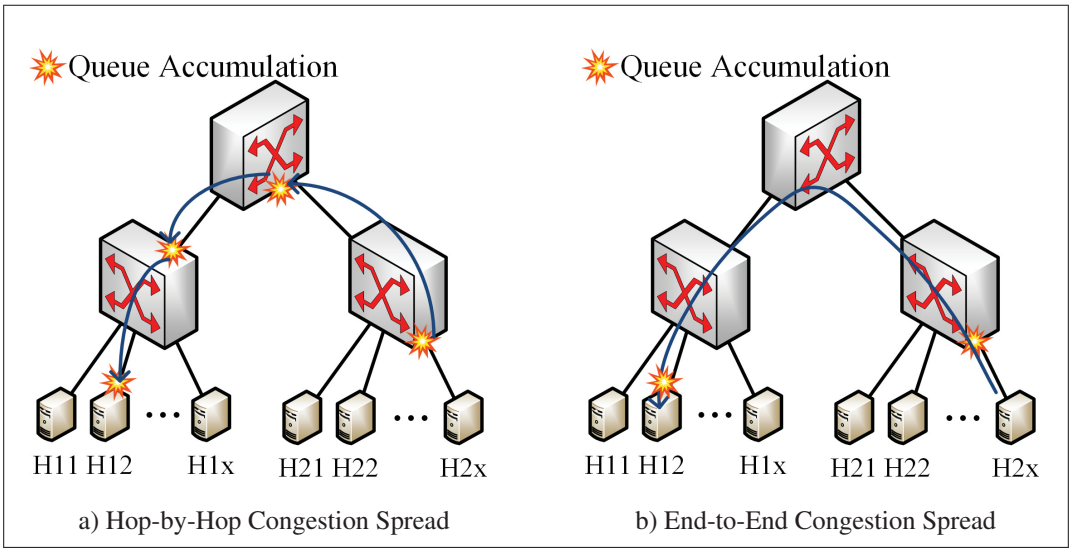


Figure 1.2 Congestion Spread Types

Conversely, end-to-end mechanisms acknowledge the source responsible for congestion directly when congestion occurs (Figure 1.2b). This involves relatively high delay until the source response. Due to this delay, hop-by-hop transport achieves considerably faster reaction

time with short-lived flows. However, due to hop-by-hop techniques limitation, namely scalability and HOL blocking, end-to-end mechanisms are preferable to control long-lived flows.

Ethernet PAUSE is a hop-by-hop congestion control mechanism. It was issued to solve the congestion problem by sending a PAUSE request to the sender when the receiver buffer reaches a specific threshold. The sender stops sending any new frames until a resume notification is received or a local timer expires. Some data flows are very sensitive to frame loss such as FCoE and iSCSI, others depend on higher layer traffic control. In addition, Ethernet PAUSE is a coarse-grained protocol because it reacts per port which causes HOL blocking.

PFC was introduced as a fine-grained protocol to mitigate the HOL blocking by enabling the operator to discriminate flows based on traffic classes that are defined in IEEE 802.1p task group (Ek, 1999). PFC divides data path into eight traffic classes; each could be controlled individually. Yet, PFC is still limited because it operates on port plus traffic class (priority) level which can cause tree saturation (Hanawa *et al.*, 1996) and HOL blocking (Stephens *et al.*, 2014).

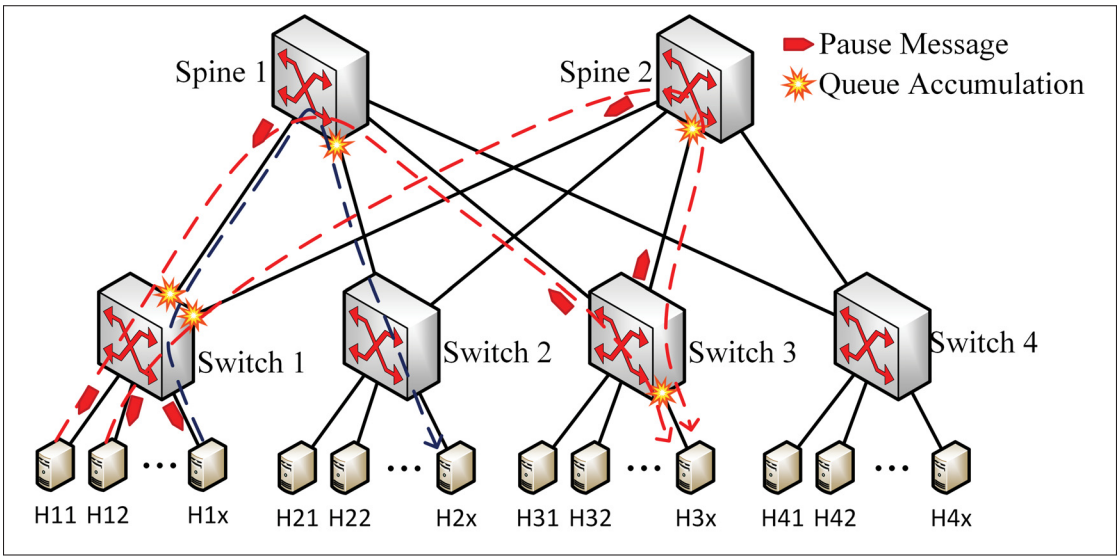


Figure 1.3 PFC HOL Blocking

Figure 1.3 shows a test case that explains the HOL blocking and congestion spreading in PFC. In this scenario, hosts H11 and H12 are sending data to host H3x and host H1x to H2x. Switch 1 executes Equal-cost multi-path routing (ECMP) and distribute the traffic on both spine 1 and spine 2. The traffic destined to H3x causes congestion at switch 3 at the output port that is connected to H3x. Switch 3 reacts by sending pause messages for all adjacent switches/ hosts that transmit data to this port (spine 1 and spine 2). The same process is repeated at spine 1 and spine 2 where two pause messages are sent to switch 1 on both its upward connections. As a final step, switch 1 reacts by sending pause messages to all adjacent nodes that send traffic to spine 1 & spine 2. It is clearly shown that PFC spreads the congestion over all the network causing what is known as tree saturation (Hanawa *et al.*, 1996) or congestion spreading. In addition, traffic that is originated from host H1x and destined to H2x is throttled at switch 1 and spine 1 due to a congestion that is originally not in its path. This phenomenon is called HOL blocking.

To ensure the maximum performance of PFC all devices have to support it, and strict buffer and timing requirements must be applied to prevent packet loss. Figure 1.4 depicts PFC buffer requirement with respect to link length. When a pause message is sent to the adjacent node, the congested queue keeps building up while the pause message is propagated through the link. To guarantee packet delivery, the propagation time of the pause message to the previous node must not exceed the time to reach the maximum buffer size (Figure 1.4). Hence the selection of buffer threshold and the length of links between every two hops are critical for PFC.

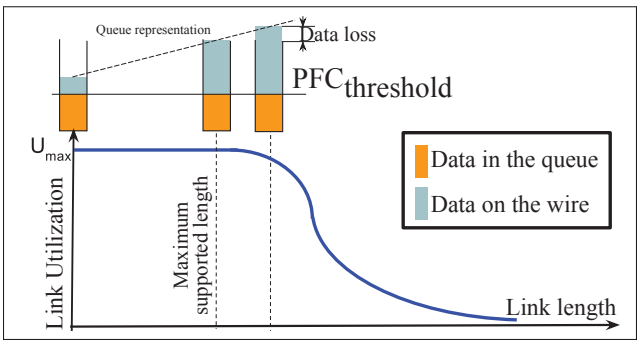


Figure 1.4 PFC Buffer Limitation (Cisco Systems, 2009)

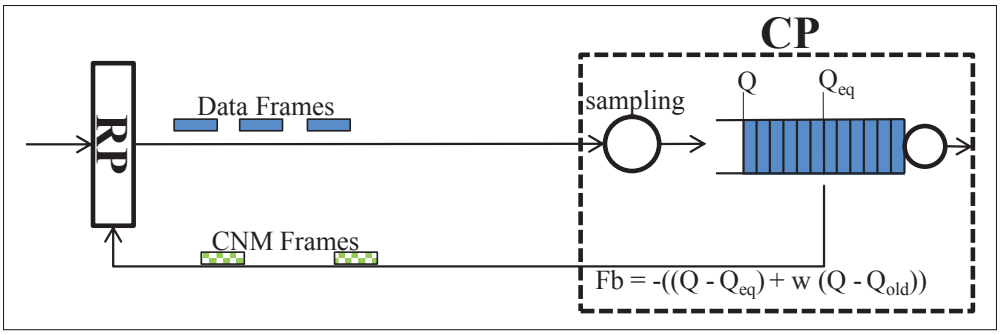


Figure 1.5 QCN framework: CP in the bridge, and RP in the host's NIC

QCN (IEEE 802.1Qau, 2010) is an end-to-end control mechanism that aims to keep queue length at a predefined level called equilibrium queue length (Q_{eq}). QCN consists of two parts, (i) Congestion Point (CP) (in bridges) and (ii) Reaction Point (RP) (in hosts) (Fig. 1.5). The CP measures queue length (Q), and calculates feedback (Fb) value, in a probabilistic manner, to reflect congestion severity (Equation 1.1).

$$Fb = -((Q - Q_{eq}) + w \times (Q - Q_{old})). \tag{1.1}$$

Where Q_{old} is the previous queue length, and w is a constant that is taken to be equal to 2 (for more details refer to (IEEE 802.1Qau, 2010)). If the calculated Fb is negative, CP creates a Congestion Notification Message (CNM) and sends it to the CP.

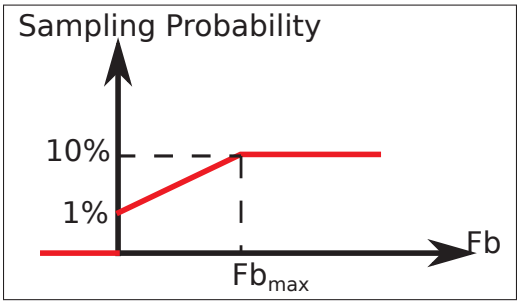


Figure 1.6 Sampling probability in QCN (Alizadeh *et al.*, 2008)

Fig. 1.6 illustrates the probability function on which QCN samples queue length and calculates Fb value as a function of the last calculated Fb.

At the end host level, when CP receives a CNM, it decreases its transmission rate accordingly. If no CNMs are received, the CP increases its transmission rate according to a three-phase rate increase algorithm (IEEE 802.1Qau, 2010).

Due to the probabilistic manner of calculating Fb, QCN experiences several issues regarding fairness (Kabbani *et al.*, 2010; Zhang & Ansari, 2013) and queue length fluctuation (Tanisawa & Yamamoto, 2013).

Moreover, both PFC and QCN functionalities are deeply integrated into switch ASICs that requires costly switch modification which we aim to avoid.

Other non-standard congestion control mechanisms are used in proprietary networks such as Credit-based Flow Control (CBFC). CBFC (Bloch *et al.*, 2011; Katevenis, 1997) is also known as back-pressure or hop-by-hop window. CBFC permits data transmission only if the transmitter knows that the downstream bridge/ host has enough buffer space. It is created originally for Virtual Channel (VC) flow control for Asynchronous Transfer Mode (ATM) network. It is still under development for Ethernet, and yet has not been standardized. Figure 1.7 depicts how CBFC operates which occurs as follow:

- Step 0: the sender starts by initializing the CBFC transmission
- Step 1: the sender requests for credit from the receiver.
- Step 2: the receiver calculates the amount that it can grant to this sender, then it sends the reply with the granted credit.
- Step 3: the sender sends data while decrementing the credit counter.
- Step 4: as the receiver receives and processes packets, it re-sends new credit to the sender.
- Step 5: step 2 to 4 is repeated until the transmission ends.

CBFC has strict buffer requirements to guarantee packet delivery. This buffer space required at the receiver is equal to the link prorogation speed multiplied by the RTT of data and credits. The buffer space, which calculated before, determine the number of packet and credit that can be transmitted over the link. To sustain the maximum link utilization, the sender must have enough packets available on his buffer and enough credit to use. The lake of one of those could

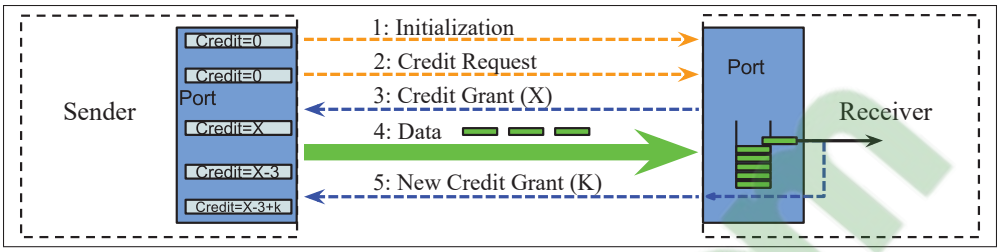


Figure 1.7 Credit-based Flow Control operating mechanism

cause link under-utilization. When traffic from many sources are sharing the same buffer, and they are not uniform, this indiscriminate sharing causes a head-of-line blocking in input queuing (Tranter *et al.*, 2007).

Few centralized solutions are proposed in the literature; e.g., Fastpass is proposed to use a centralized arbiter to packet transmission (Perry *et al.*, 2014). Instead of using the current network architectures which distribute packet transmission decisions among the hosts and path selection decisions among network switches, the arbiter controls tightly both packet transmission and path selection. Each host is extended as in figure 1.8 by adding Fastpass Control Protocol (FCP). When host's applications send data to the network card, it is interrupted by the FCP. FCP sends this demand in a request message to the Fastpass arbiter, specifying the destination and data size (Figure 1.8). The arbiter allocates a set of time slots for this data packets, and determine the path that is used by these packets. The arbiter keeps track of time-slots assignment history in its database. Based on this previous reservation, the arbiter can determine time-slots and path availability for new requests.

Another approach that exists in the literature is using the measure or estimate end-to-end available bandwidth. Due to the difficulty of measuring the available bandwidth in real time, few articles address flow control mechanisms based on this approach. Forward Explicit Congestion Notification (FECN) mechanism is presented in (Jiang *et al.*, 2008). In FECN, sources periodically send probe packets. The switches along the path modify these packets with the available bandwidth. Once they reach their destination, they are reflected back to the source. Then the source reacts according to the available bandwidth information that exists in the probe pack-

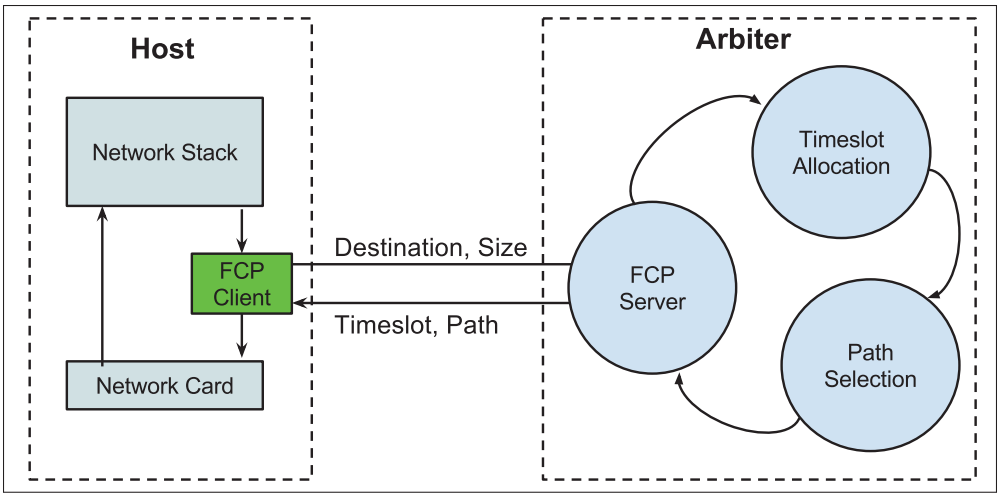


Figure 1.8 Fastpass Arbiter Architecture

ets. Sending feedback directly from the switches to the source in case of severe congestion is proposed in (So-In *et al.*, 2008) to enhance the performance of FECN.

(Jeyakumar *et al.*, 2014) proposes the use of tiny packets that include programs and extending the switches to forward and execute these tiny packet programs (at most 5 instructions) at line speed. In addition, It proposes to extend the end-hosts to perform arbitrary computation on network state that is retrieved from the switches. The authors use this proposition to address many issues; congestion control is one of them. Yet, implementing these mechanisms requires modifying Ethernet switches.

1.1.1.2 Transmission layer congestion control protocols

A vast amount of research is done to enhance TCP protocol in order to reduce queueing delay in data center networking such as Brakmo *et al.* (1994); Alizadeh *et al.* (2010, 2012); Zhu *et al.* (2015); Wilson (2008) and (Ha *et al.*, 2008). However, it is commonly known that TCP under-utilizes network with high Bandwidth Delay Product (BDP). Therefore, due to the vast increase in Ethernet bandwidth, TCP performance decreases.

As it was shown mathematically that TCP utilization is equal to 75% in average, TCP Vegas was proposed in (Brakmo *et al.*, 1994) to improve TCP throughput. TCP Vegas aims

to eliminate congestive losses and to increase the bottleneck link utilization. TCP Vegas main contribution is measuring RTT with finer granularity and using this measurement to detect congestion at an incipient stage. TCP Vegas monitor the difference between the actual throughput (calculated using the measured RTT) and the maximum throughput (calculated using the minimum RTT of all measured round-trip times usually measured at the beginning of the session). TCP Vegas is based on the idea that the number of bytes in transit is directly proportional to the expected throughput. Therefore, it is expected that the throughput increases as the window size increases. TCP Vegas calculate $Diff$ as the difference between the expected threshold and the actual threshold. $Diff$ represents the extra data that should not have sent if the used bandwidth matches the available bandwidth. TCP Vegas compares $Diff$ with two thresholds α and β that represent a lower and a higher boundary respectively. Based on these thresholds, TCP Vegas increases the congestion window linearly if $Diff < \alpha$. If $Diff > \beta$, TCP Vegas decreases the congestion window linearly. TCP Vegas leaves the congestion window unchanged if $\alpha < Diff < \beta$ as shown in Fig. 1.9.

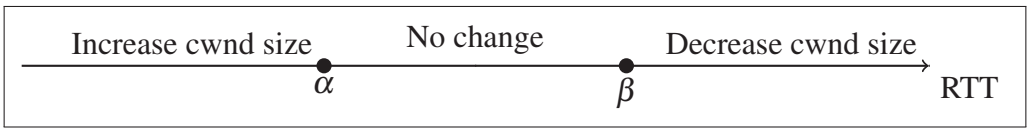


Figure 1.9 TCP Vegas operating modes

Due to this early reaction of TCP Vegas, if a TCP Vegas-controlled traffic shares a network with other TCP variant traffic, Vegas-controlled traffic gets throttled and faces fairness issue. E.g. TCP Vegas is considered non-TCP friendly protocol; therefore, it is not widely used.

TCP protocol reacts to congestion upon packet loss. Thus, Data Center TCP (DCTCP) was proposed as a data center variant of TCP to avoid loss-based reaction of TCP (Alizadeh *et al.*, 2010). DCTCP uses ECN marking to detect congestion and reacts accordingly. DCTCP uses a marking scheme at ECN-capable switches that set the ECN bit of packets once the buffer occupancy exceeds a fixed threshold. Instead of dropping the window size in half like tradi-

tional TCP, DCTCP reacts in proportion to the extent of congestion. DCTCP source reacts by reducing the window by a factor of the fraction of marked packets.

Trading a little bandwidth in order to achieve low queue length and low latency is discussed in a number of papers. For example, High-bandwidth Ultra-Low Latency (HULL) (Alizadeh *et al.*, 2012) is presented to reduce average and tail latencies in data center network by sacrificing a small amount of bandwidth (e.g., 10%). HULL presents the Phantom Queue (PQ) as a new marking algorithm based on link utilization rather than queue occupancy (by setting ECN bit). The challenges of HULL are the need for switch modification.

Both TCP Vegas and DCTCP kept the original TCP window-based behavior. Due to the rapid increase in the control cycle time, defined mainly by propagation delay compared to transmission time in modern networks, window-based schemes encounter significant challenges (Charny *et al.*, 1995). Thus, few congestion control mechanisms advocate rate-based schemes. For example, Data Center QCN (DCQCN) (Zhu *et al.*, 2015) tries to combine the characteristics of DCTCP and QCN in order to achieve QCN-like behavior while using the Explicit Congestion Notification (ECN) marking feature that is available in ECN-aware switches. However, extensive experiments are presented in (Mittal *et al.*, 2015) which shown that ECN-based congestion signal does not reflect the queue state. Conversely, delay correlates strongly with queue buildups in the network. Therefore, in our research, we build our proposal on a delay-based concept.

TIMELY (Mittal *et al.*, 2015) is a delay-based congestion control scheme for data centers. It uses the deviation of RTT to identify congestion. TIMELY relies on the capability of NIC hardware to obtain fine-grained RTT measurements. In TIMELY, the receiver sends an acknowledge per data segment of size 16 - 64 KB. At the sender, upon receiving an ACK, RTT is calculated and the gradient of RTT in order to control the transmission rate. TIMELY defines RTT as the propagation and queuing delay only. Thus, segment serialization time (time to put the segment on the wire) is subtracted from the completion time in order to calculate RTT as shown in Fig. 1.10.

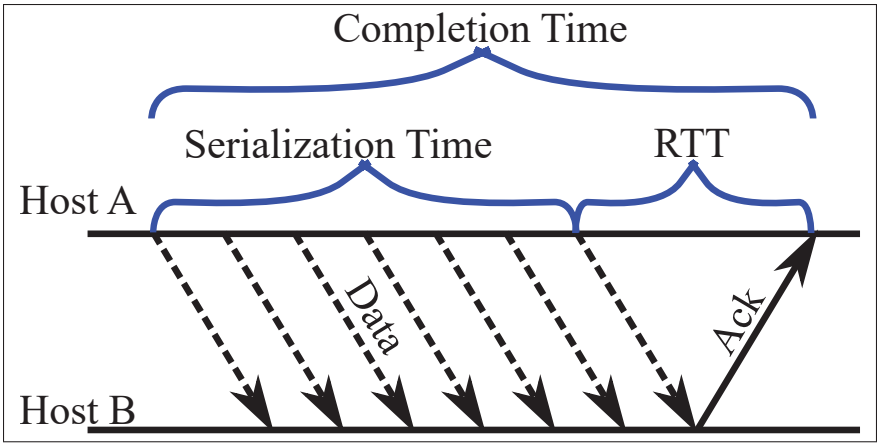


Figure 1.10 RTT calculation in TIMELY

TIMELY is a rate-based protocol that computes a new rate after receiving each ACK based on RTT value as follow:

- If RTT is less than T_{low}, TIMELY increases transmission rate R additively by a constant δ.
- If RTT is greater than T_{high}, TIMELY decreases R multiplicatively by a factor β.
- if RTT is between T_{low} and T_{high}, TIMELY calculates RTT gradient $g = \frac{RTT - RTT_{old}}{D_{minRTT}}$ and controls the transmission rate using (1.2).

$$\begin{cases} R \leftarrow R + \delta & \text{If } RTT < T_{low} \\ R \leftarrow R \times (1 - \beta \frac{T_{high}}{RTT}) & \text{If } RTT > T_{high} \\ \begin{cases} R \leftarrow R + N \times \delta & \text{If } g \leq 0 \\ R \leftarrow R \times (1 - \beta \times g) & \text{If } g > 0 \end{cases} & \text{Otherwise} \end{cases} \tag{1.2}$$

TIMELY uses per-packet pacing to apply the newly calculated rate and uses a delay-based technique to detect congestion. Mathematical and heuristic comparisons between TIMELY and our proposed solution are presented in Chapter 3.

1.1.1.3 CPRI over Ethernet Challenges

Furthermore, we study the possibility of using Ethernet network to provide a transport network for fronthaul in 5G network. The exponential increase in mobile network users and the enormous bandwidth required by new mobile applications lead to massive increase in mobile data traffic. It is anticipated that by 2021 smartphone subscriptions will double to 6.4 billion subscriptions exchanging 1.6 Zettabytes of data (Ericsson, 2016). These characteristics require the envisioned 5G mobile networks to provide very high rates (up to 10 Gbps per user) and submilliseconds latency, particularly for time-critical applications. To achieve ultra-high user data rates, 5G networks require higher transmission frequencies which lead to shorter radio transmission distance. This could be achieved by distributing the Remote Radio Heads (RRHs) into smaller cells.

A promising approach to reconcile these requirements, with conservative investments, is to split the mobile network node into REC (i.e. a baseband unit (BBU) which processes baseband signals and is located in a central office) and the RE (i.e. RRHs that are distributed in each cell and consist of an antenna and basic radio functionality).

Originally, this solution was called Centralized Radio Access Network (C-RAN) since many lightweight RRHs are deployed in smaller cells and connected to fewer BBUs in a centralized BBU pool. The emergence of virtualization and cloud computing with its cost efficiency, high performance, scalability, and accessibility led to a novel approach that virtualizes the BBU pool in the cloud. Therefore, the solution name changed from centralized RAN to cloud RAN C-RAN (Mobile, 2013). Moreover, an analysis on statistical multiplexing gain is performed in (Namba *et al.*, 2012). The analysis shows that in Tokyo metropolitan area, the number of BBUs can be reduced by 75% compared to the traditional RAN architecture. Further, virtualized RECs can move across different REC pools according to traffic/load requirements. Tidal effect is an example that shows the advantages of this virtualized proposal. Base stations are often dimensioned for busy hours, and users move between cells. Thus, in a given period when users move, for example, from office to residential areas, a huge amount of processing power

is wasted in the regions where the users have left. By moving the digital processing units into a centralized location, network resources (in this case a BBU pool) could be allocated/deallocate based on traffic load. Consequently, it increases network efficiency and reduces cost (Checko, 2016).

In C-RAN, the separation between REC and RE introduces the Fronthaul network as shown in Fig. 1.11. This fronthaul network is responsible for carrying digitized complex In-phase and Quadrature-phase (I/Q) radio samples between the RRHs and the BBUs.

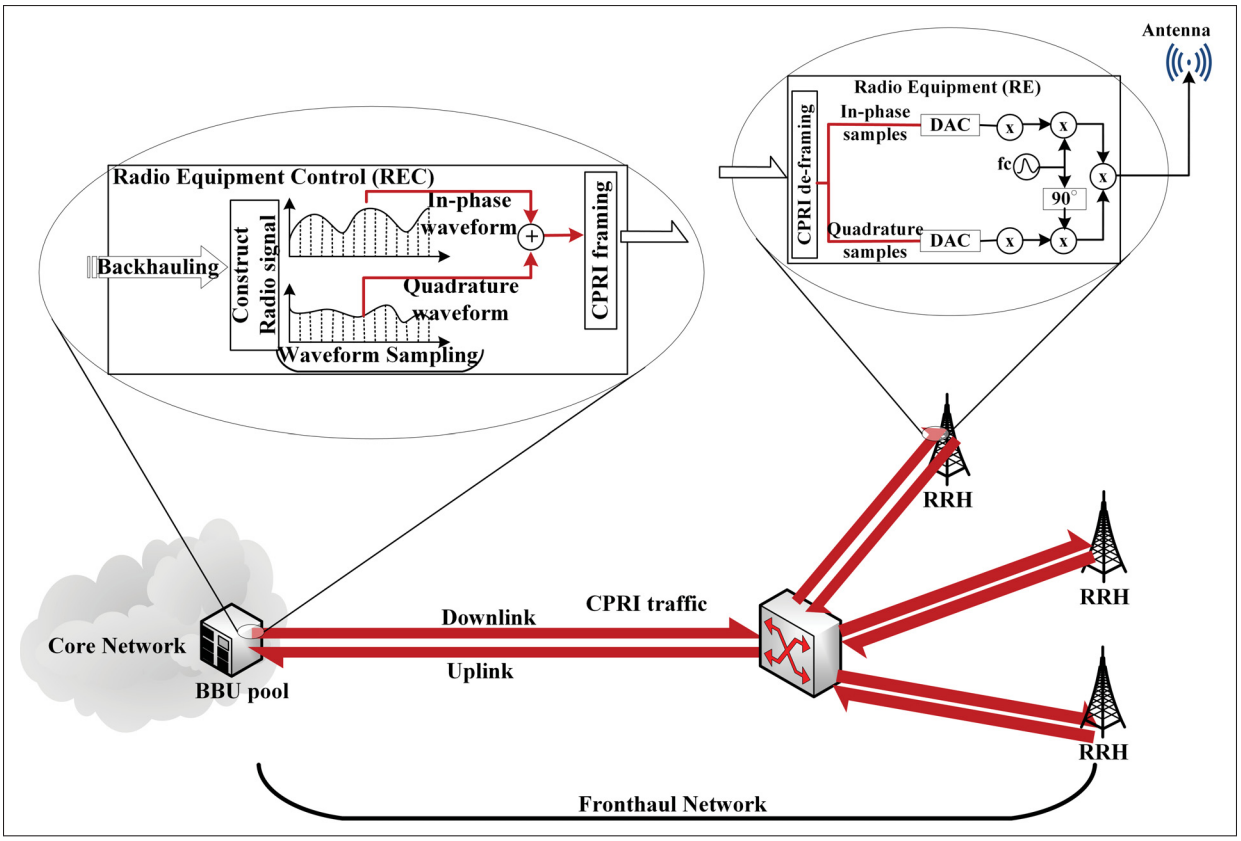


Figure 1.11 C-RAN architecture.

Several ongoing projects, such as Time-Sensitive Networking for Fronthaul IEEE 802.1CM (Institute of Electrical & Electronic Engineers, 2017a), Packet-based Fronthaul Transport Networks IEEE P1914.1 (Institute of Electrical & Electronic Engineers, 2017b) and RoE Encapsulations and Mappings IEEE P1914.3 (Institute of Electrical & Electronic Engineers, 2017c)

strive to define an interface between REC and RE. CPRI (Ericsson AB, Huawei Technologies Co. Ltd, NEC Corporation, Alcatel Lucent, and Nokia Networks, 2015) is defined as the internal interface between REC and RE. CPRI is designed based on the digital radio over optical fiber concept where the radio signal is sampled, quantized and transmitted over optical networks. However, optical networks could be cost inefficient in some scenarios; e.g., building an optical network to connect RRHs, that are distributed in skyscraper floors, is cost-inefficient. Whereas, building an Ethernet network or using existing networks introduces huge cost reduction. Therefore, a cost-efficient, flexible and re-configurable mobile fronthaul that supports emerging network paradigms becomes imperative.

Transporting CPRI over Ethernet network has recently drawn the attention of both the industry and the academia because of its cost efficiency. Ethernet network is widely used in access and data-center networks. It has also shown huge capacity growth lately. Accordingly, encapsulating CPRI traffic over Ethernet introduces significant savings in CAPEX and OPEX. Furthermore, the OAM capabilities of Ethernet provide standard methods for network management and performance monitoring. However, CPRI traffic has very stringent requirements regarding jitter, latency, bandwidth, Bit Error Rate (BER), and network synchronization that must be satisfied by the transport network. Therefore, in Chapter 4, we provided a solution that supports transporting CPRI traffic over Ethernet network.

1.2 Summary of Publications

In addition to the articles presented in the three ensuing chapters, this research has produced the following publications.

1.2.1 Using Ethernet commodity switches to build a switch fabric in routers

This paper is published in the proceedings of in IEEE Computer Communication and Networks (ICCCN), 2015 (Bahnasy *et al.*, 2015).

In this paper, we tackle the congestion control for switch fabric in routers. We propose Ethernet Congestion Control & Prevention (ECCP), as a novel concept to control and prevent congestion in switch fabrics. ECCP controls congestion by preventing hosts from exceeding their bandwidth fair share. To evaluate the performance of ECCP, we conduct a simulation model using OMNEST simulator. Our analysis confirms that ECCP is a viable solution to (1) avoid congestion within the fabric, thus minimizing path latency and avoiding packet loss, (2) guarantee the fair share of link capacity between flows, and (3) avoid HOL blocking.

1.2.2 Proactive Ethernet Congestion Control Based on Link Utilization Estimation

This paper is published in the proceedings of in IEEE International Conference on Computing, Networking and Communications (ICNC), 2016 (Bahnasy *et al.*, 2016).

In this paper, we propose No-Probe ECCP (NoP-ECCP) as enhancements for the algorithm used by ECCP to reduce probe packet overhead. In this variant of ECCP we present a new mechanism to control host transmission rates based on link utilization estimation instead of available bandwidth estimation. The results obtained through simulations show that NoP-ECCP outperforms ECCP and QCN in terms of fairness, link utilization and queue length.

1.3 METHODOLOGY

In order to accomplish the research goals, our methodology involves the development of an accurate model of Ethernet in a simulator. This model is processed on these progressive stages:

- Build a base model of Ethernet network.
- The base model is compared against a lab environment to validate the simulator.
- The base model is augmented with several standard congestion mechanisms and the basic behavior of these mechanisms is verified.
- The base model is augmented with our proposed congestion mechanism.

- A number of scenarios is simulated in order to answer the questions mentioned on the objective section and refine the mechanism.
- A lab environment will also be setup where we will verify the implementation of the congestion prevention mechanism and compare the results against the simulator.
- Repeat steps 1 - 4 with other network topologies and different scenarios (network and hosts configuration).

1.3.1 Ethernet congestion control and prevention

This patent is published in the US patent office with the number PCT/IB2016/050,738 (Beliveau *et al.*, 2016).

In this publication, Ericsson Canada is protecting its proprietary rights by filing a patent for ECCP protocol that was published in (Bahnasy *et al.*, 2015) and (Bahnasy *et al.*, 2016).

1.3.2 HetFlow: A Distributed Delay-based Congestion Control for Data Centers to Achieve ultra Low Queueing Delay

This paper is published in the proceedings of in IEEE International Conference on Communications (ICC), 2017 (Bahnasy *et al.*, 2017).

In this paper, we explore the possibility of controlling congestion in data centers while guaranteeing no packet loss, fairness, no head-of-line blocking, and low latency. We propose HetFlow (Heterogeneous Flow) as an Ethernet delay-based congestion control mechanism that controls congestion while achieving minimum queue length, minimum network latency, and high link utilization. In addition, HetFlow was designed to guarantee fairness between flows with different packet sizes and different round-trip times (RTTs). The results obtained through prototype and simulations show that HetFlow succeeded in preventing congestion and achieving low queue length, high link utilization, and fairness between flows.

1.3.3 Heterogeneous Flow Congestion Control

This patent is submitted to the US provisional patent office with serial number 62/408.363 filed on October 2014.

In this publication, Ericsson Canada is protecting its proprietary rights by filing a patent for HetFlow protocol that was published in (Bahnasy *et al.*, 2017).

1.3.4 CPRI over Ethernet: Towards fronthaul/backhaul multiplexing

This paper is published in the proceedings of in IEEE Consumer Communications & Networking Conference (CCNC), 2018 (Bahnasy *et al.*, 2018a).

Ethernet has been proposed for the 5G fronthaul to transport the Common Public Radio Interface (CPRI) traffic between the radio equipment (RE) and the radio equipment control (REC). In this paper, we introduce distributed timeslot scheduler for CPRI over Ethernet (DTSCoE) as a scheduling algorithm for IEEE 802.1Qbv to support CPRI traffic. DTSCoE is built upon the stream reservation protocol (SRP) IEEE 802.1Qcc to propagate timeslot information across the datapath without any centralized coordination. The simulation results demonstrate that DTSCoE reduces one-way delay to minimum and reduces the jitter to zero which satisfies the CPRI requirements.

1.3.5 DTSRPoE - Distributed Time-Slot Reservation Protocol over Ethernet

This patent is under process to be submitted to the US provisional patent office with Ericsson internal number P71707.

In this publication, Ericsson Canada is protecting its proprietary rights by filing a patent for DTSRPoE protocol that was published at (Bahnasy *et al.*, 2018a).

CHAPTER 2

ZERO-QUEUE ETHERNET CONGESTION CONTROL PROTOCOL BASED ON AVAILABLE BANDWIDTH ESTIMATION (Bahnasy *et al.*, 2018b)

Mahmoud Bahnasy[1], Halima Elbiaze[2], Bochra Boughzala[3]

[1] Département de Génie électrique, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Département d'informatique, Université du Québec à Montréal

[3] Ericsson Canada, bochra.boughzala@ericsson.com

This article was accepted for publication at «Elsevier International Journal of Computer and Telecommunications Networking» in December 2017.

2.1 Abstract

Router's switch fabric has strict characteristics in terms of packet loss, latency, fairness and head-of-line (HOL) blocking. Network manufacturers address these requirements using specialized, proprietary and highly expensive switches. Simultaneously, IEEE introduces Data Center Bridging (DCB) as an enhancement to existing Ethernet bridge specifications which include technological enhancements addressing packet loss, HOL blocking and latency issues. Motivated by DCB enhancements, we investigate the possibility of using Ethernet commodity switches as a switch fabric for routers. Thereby, we present Ethernet Congestion Control Protocol (ECCP) that uses Ethernet commodity switches to achieves flexible and cost-efficient switch fabric, and fulfills the strict router characteristics. Furthermore, we present a mathematical model of ECCP using Delay Differential Equations (DDEs), and analyze its stability using the phase plane method. We deduced the sufficient conditions of the stability of ECCP that could be used for parameter setting properly. We also discovered that the stability of ECCP is mainly ensured by the sliding mode motion, causing ECCP to keep cross traffic close to the maximum link capacity and queue length close to zero. Extensive simulation scenarios are driven to validate the analytical results of ECCP behavior. Our analysis shows that ECCP is

practical in avoiding congestion and achieving minimum network latency. Moreover, to verify the performance of ECCP in real networks, we conducted a testbed implementation for ECCP using Linux machines and a 10-Gbps switch.

2.2 Introduction

Router's switch fabric is an essential technology that is traditionally addressed using custom Application-Specific Integrated Circuit (ASIC). This ASIC must fulfill particular characteristics including low packet loss, fairness between flows, and low latency (Bachmutsky, 2011). The emergence of very-high-speed serial interfaces and new router's architectures increase the design and manufacturing cost of the switch fabric chipset. Traditionally, switch fabric is manufactured using either shared memory or crossbar switch as shown in Fig. 2.1a and Fig. 2.1b respectively. The shared memory architecture requires memory that works N times faster than port speed, where N is the number of ports which raises scalability issue. On the other hand, crossbar architecture tries to keep the buffering at the edge of the router (Virtual Output Queue VOQ inside line cards). Because this architecture requires N VOQs at each ingress port and a central unit (arbiter), it faces scalability issue (Lee, 2014).

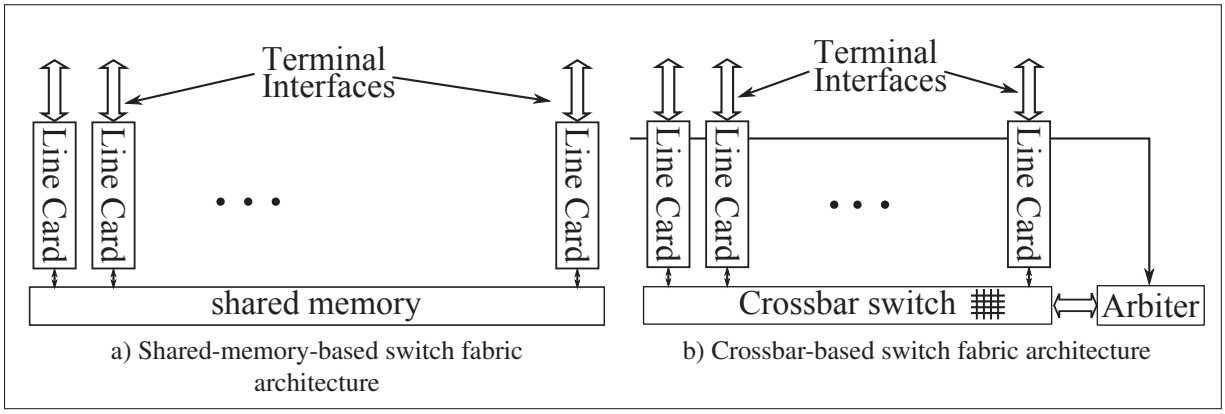


Figure 2.1 Router's switch fabric architectures

In this research, we introduce a new router architecture that uses Ethernet commodity switches as a switch fabric. In this architecture, we keep all buffering at the edge of the router and an

Ethernet switch is used as a switch fabric. IEEE has recently presented Data Center Bridging (DCB) (802.1, 2013) that comprises several enhancements to Ethernet network. However, Ethernet network still suffers from HOL blocking, congestion spreading and high latency. To overcome these limitations and achieve a non-blocking switch fabric, we present Ethernet Congestion Control Protocol (ECCP) that maintains Ethernet network non-blocked by preserving switches' queue lengths close to zero leading to minimum latency and no HOL blocking. Unlike traditional Congestion control mechanisms that use packet accumulation in buffers to trigger the rate control process, ECCP estimates available bandwidth (*AvBw*) and uses this information to control transmission rates.

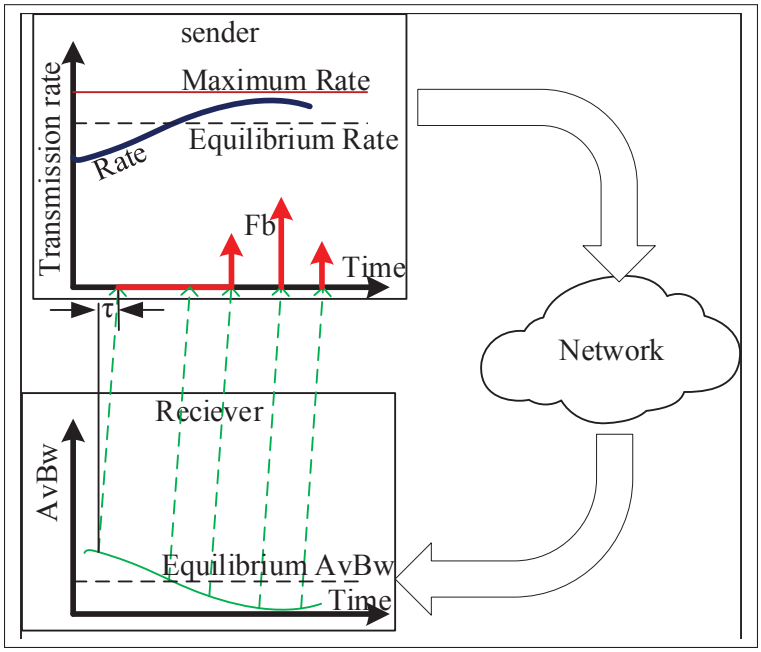


Figure 2.2 ECCP overview

Fig. 2.2 illustrates a logical overview of how ECCP estimates Available Bandwidth *AvBw* compared to the desired equilibrium rate at the receiver side and convey this information to the sender. Consequently, the sender can calculate a feedback Fb value that is used to control the transmission rate. One can notice that, Fb is equal to zero for *AvBw* that is greater than the equilibrium point and positive otherwise. Therefore, when *AvBw* is less than the equilibrium rate, the sender starts rate throttling before link saturation or data accumulation in the queues.

Accordingly, ECCP achieves minimum latency by trading off a small margin of link capacity. Hence, ECCP achieves (i) ultra low queue length (close-to-zero level), (ii) low latency, and (iii) high throughput, (iv) with no switch modification.

Such a mechanism could be used in manufacturing a cost-efficient routers' switch fabric while guaranteeing traditional router characteristics. Besides, it can be utilized as a reliable and robust layer 2 congestion control mechanism for data center applications (e.g. high-performance computing (Snir, 2014), remote direct memory access (RDMA) (Bailey & Talpey, 2005), and Fibre Channel over Ethernet (FCoE) (Kale *et al.*, 2011)).

Furthermore, we introduce a mathematical model of ECCP while using the phase plane method. First, we build a fluid-flow model for ECCP to derive the delay differential equations (DDEs) that represent ECCP. Then, we sketch the phase trajectories of the rate increase and rate decrease subsystems. Consequently, we combine these phase trajectories to understand the transition between ECCP's subsystems and to obtain the phase trajectory of the global ECCP system. Subsequently, the stability of ECCP is analyzed based on this phase trajectory. Our analysis reveals that the stability of ECCP depends mainly on the sliding mode motion (Utkin, 1977). Thereafter, we deduce stability conditions that assist in defining proper parameters for ECCP. Besides, several simulations are conducted using OMNEST (Varga & Hornig, 2008) to verify our mathematical analysis. Finally, a Linux-based implementation of ECCP is conducted to verify ECCP's performance through experiment.

The rest of this paper is organized as follows. Related work is introduced in Section 2.3. Section 2.4 presents ECCP mechanism. Section 2.5 introduces the phase plane analysis method in brief. The mathematical model of ECCP is derived in Section 2.6. The stability analysis of ECCP is deduced in Section 2.7. Linux-based implementation is presented in Section 2.8. Finally, Section 2.9 introduces conclusion and future work.

2.3 Related Work

In this section, we present some research work that is closely related to congestion control in both Ethernet layer and Transmission Control Protocol (TCP) layer. IEEE has recently presented Data Center Bridging (DCB) (802.1, 2013) that comprise several enhancements for Ethernet network to create a consolidation of I/O connectivity through data centers. DCB aims to eliminate packet loss due to queue overflow. Ethernet PAUSE IEEE 802.3x and Priority-based Flow Control (PFC) (IEEE Standard Association, 2011) are presented in DCB as link level (hop-by-hop) mechanisms. Ethernet PAUSE was issued to solve packet loss problem by sending a PAUSE request to the sender when the receiver buffer reaches a certain threshold. Thus, the sender stops sending data until a local timer expires or a resume notification is received from the receiver. PFC divides data path into eight traffic classes, each could be controlled individually. Yet, PFC is still limited because it operates on port plus priority level which can cause congestion-spreading and HOL blocking (IEEE Standard Association, 2011; Stephens *et al.*, 2014).

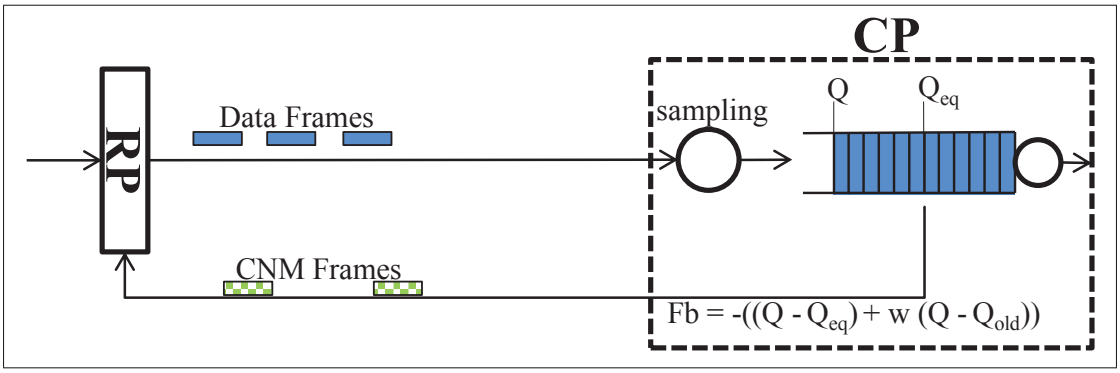


Figure 2.3 QCN framework: CP in the bridge, and RP in the host's NIC

Quantized Congestion Notification (QCN) (IEEE 802.1Qau, 2010; Alizadeh *et al.*, 2008) is an end-to-end control mechanism which is standardized in IEEE 802.1Qau (IEEE 802.1Qau, 2010). QCN aims to keep queue length at a predefined level called equilibrium queue length (Q_{eq}). QCN consists of two parts, (i) Congestion Point (CP) (in bridges) and (ii) Reaction Point (RP) (in hosts) (Fig. 2.3). The CP measures queue length (Q), and calculates a feedback (Fb)

value, in a probabilistic manner, to reflect congestion severity (Equation 2.1).

$$Fb = -((Q - Q_{eq}) + w \times (Q - Q_{old})). \tag{2.1}$$

Where Q_{old} is the previous queue length and w is a constant which equals 2 (for more details refer to (IEEE 802.1Qau, 2010)). If the calculated Fb is negative, CP creates a Congestion Notification Message (CNM) and sends it to the RP.

QCN reduces the overhead of control information traffic and reduces the required computational power by calculating Fb in a probabilistic manner. At the end host, when RP receives CNM, it decreases its transmission rate accordingly. If no CNM is received, the RP increases its transmission rate according to a three-phase rate increase algorithm (IEEE 802.1Qau, 2010).

Due to the probabilistic manner of calculating Fb, QCN experiences several issues regarding fairness (Kabbani *et al.*, 2010; Zhang & Ansari, 2013) and queue length fluctuation (Tanisawa & Yamamoto, 2013). In addition, QCN does not achieve minimum latency as it keeps queue length at a certain level (Q_{eq}).

Several research papers have discussed various enhancements for QCN. For example, (Tanisawa & Yamamoto, 2013) presents the use of delay variation as an indication of congestion to address queue fluctuation issue. Other studies like (Kabbani *et al.*, 2010; Zhang & Ansari, 2013) addressed QCN fairness issue by using new Active Queue Management (AQM) (Aweya *et al.*, 2001) algorithms that are capable of identifying the culprit flows. Thus, they send CNMs for each culprit flow. These techniques achieve fairness but they are implemented in the switch which we aim to avoid.

Data Center TCP (DCTCP) (Alizadeh *et al.*, 2010) uses switches that support Explicit Congestion Notification (ECN) to mark packets that arrive while queue length is greater than a predefined threshold. DCTCP source reacts by reducing the window proportionally to the fraction of marked packets. Data Center QCN (DCQCN) (Zhu *et al.*, 2015) combines the characteristics of Data Center TCP (DCTCP) (Alizadeh *et al.*, 2010) and QCN in order to achieve QCN-

like behavior while using the ECN marking feature. DCQCN requires very strict parameters selection regarding byte counter and marking probability.

Trading a little bandwidth to achieve low queue length and low latency is discussed in a number of papers. For example, HULL (High-bandwidth Ultra-Low Latency) is presented in (Alizadeh *et al.*, 2012) to reduce average and tail latencies in data centers by sacrificing a small amount of bandwidth (e.g., 10%). HULL presents the Phantom Queue (PQ) as a new marking algorithm. Phantom queues simulate draining data at a fraction (< 1) of link rate. This process generates a virtual backlog that is used to mark data packets before congestion. The challenges of HULL are the needs of switch modification.

TIMELY (Mittal *et al.*, 2015) is a congestion control scheme for data centers. It uses the deviation of Round-Trip Time (RTT) to identify congestion, instead of ECN marking in DCTCP. TIMELY can significantly reduce queuing delay and it would be interesting to compare ECCP and TIMELY in future work.

Enhanced Forward Explicit Congestion Notification (E-FECN) (So-In *et al.*, 2008) and proactive congestion control algorithm (PERC) (Jose *et al.*, 2015) are presented as congestion control mechanisms that exploit the measured available bandwidth to control data rates. However, these two methods require switch modifications which we aim to avoid.

Few centralized solutions are proposed in the literature. For example, Fastpass (Perry *et al.*, 2014) embraces central control for every packet transmission which raises a scalability issue.

Another approach to enhance the performance of TCP protocol was to distinguish between congestive packet loss and non-congestive packet loss (Wang *et al.*, 2016; A. *et al.*, 2017). Therefore, the TCP congestion avoidance algorithm could be activated only when congestive packet loss is detected. E.g., TCP INVS (Wang *et al.*, 2016) estimates network queue length and compare this estimation to a threshold. If the estimated queue length exceeds the threshold, the loss is caused by congestion. Consequently, TCP INVS activate the traditional congestion avoidance algorithm. Otherwise, the loss is considered as a non-congestion loss and TCP INVS

ignores it and avoids limiting congestion window growth. In addition, (A. *et al.*, 2017) proposes an RTT estimation algorithm using Autoregressive Integrated Moving Average (ARIMA) model. By analyzing the estimated RTT, one can estimates the sharp and sudden changes in the RTT, thereby differentiating the non-congestive packet loss from congestive packet loss. While these mechanisms acheive better throughput on lossy networks, it introduces an extra packet loss that is not suitable for router switch fabric or data center network.

Optimizing the routing decision to control the congestion is also proposed in several research papers. Most of this research follows a key idea called the back-pressure algorithm (Tassiulas & Ephremides, 1992) where traffic is directed around a queuing network to achieve maximum network throughput. An example of this scheme is presented in (Liu *et al.*, 2016) where the authors developed a second-order joint congestion control and routing optimization framework that aims to offer resource optimization and fast convergence. Such a scheme can significantly reduce queuing delay and it would be interesting to investigate this scheme in future work.

2.4 ECCP : Ethernet Congestion Control Protocol

In this section, we present ECCP as a distributed congestion prevention algorithm that works on Ethernet layer. ECCP controls data traffic according to the estimate Available Bandwidth (*AvBw*) through a network path. ECCP strives to keep link occupancy less than the maximum capacity by a percentage called Availability Threshold (*AvT*). Traditional congestion control mechanisms aim to keeps the queue around a target level. These mechanisms can reduce queuing latency, but they cannot eliminate it. In these mechanisms, a non-zero queue must be observed before reaction, and sources need one RTT to react to this observation, which causes data accumulation in queues even further. On the other hand, ECCP uses *AvBw* as a congestion signal to trigger sources reaction before data accumulation in the queue. Therefore, ECCP achieves a close-to-zero queue length, leading to minimum network latency.

ECCP estimates *AvBw* through network path by sending trains of probe frames periodically through this path. Sender adds sending time and other information as train identifier and sequence number within the train to each probe frame. On the receiver side, ECCP receives these frames and estimates *AvBw* using a modified version of Bandwidth Available in Real-Time (BART) (Ekelin *et al.*, 2006). Afterward, ECCP transmits this information back to the sender. At the sender side, ECCP controls transmission rate based on the received *AvBw* value. ECCP advocates rate-based control schemes instead of window-based control schemes because window-based schemes encounter significant challenges particularly with the rapid increase of the control cycle time, defined mainly by propagation delay, compared to transmission time in modern networks (Charny *et al.*, 1995). In addition, (Raina *et al.*, 2005) and (Kelly *et al.*, 2008) state that at high line rates, queue size fluctuations become fast and difficult to control because queuing delay is shorter than the control loop delay. Thus, rate based control schemes are more reliable.

2.4.1 ECCP components

In this section, ECCP architecture is described in detailed and the interactions between its components are explained. ECCP prevents congestion by keeping a percentage of the link capacity available called Availability Threshold (AvT). Thus, for any link of maximum capacity C, AvT creates a bandwidth stability margin equals $AvT \times C$. This bandwidth stability margin allows ECCP to send probe traffic without causing queue accumulation. ECCP does not require switch modification because all its functionalities are implemented inside line cards or hosts.

Fig. 2.4 depicts ECCP components[1] : (1) ***probe sender***, (2) ***probe receiver***, (3) ***bandwidth estimator***, and (4) ***rate controller***. These modules are implemented in every line card in the router or every host.

[1] The rate limiter in Fig. 2.4 is outside the scope of this paper

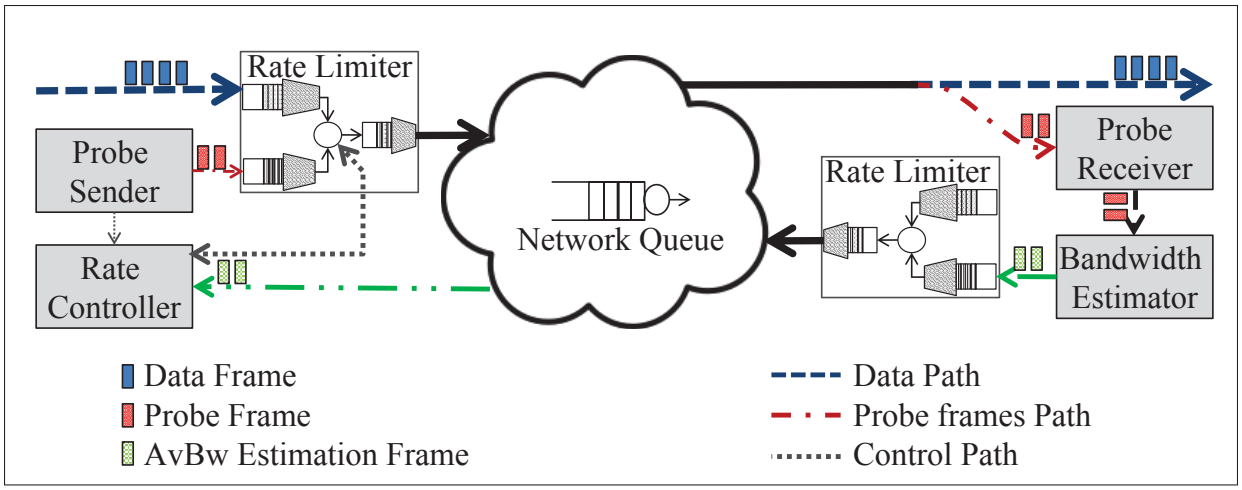


Figure 2.4 ECCP components

2.4.1.1 ECCP probe sender

ECCP control cycle starts with the ***probe sender***. This module generates probe trains each of size N frames. Thereupon, it sends them through the network towards destination host. By the time they leave the source, each probe frame is tagged with a sending time. Other information is added to the probes such as sequence number and train identifier. ECCP ***probe sender*** sends probe traffic of a uniformly distributed random rate μ between a fixed minimum value and $R \times AvT$ where R is the transmission rate. ECCP is not trying to estimate an exact value for $AvBw$. Instead, it only estimates $AvBw$ value within a maximum limit equals $R \times AvT$. Thus, ECCP gets enough information to control (decrease or increase) data rate while limiting probe rate to $R \times AvT$. Hence, probe traffic for M flows crossing one link $(M \times R \times AvT)$ never exceeds link bandwidth stability margin $(AvT \times C)$.

According to that model, the probe rate has uniform distribution. Thus, the average probe overhead for each flow equals $0.5 \times (\text{minimum probe rate} + AvT \times R) \approx 0.5 \times AvT \times R$. Thus, the probe overhead for all flows $X_{probes} \approx 0.5 \times AvT \times \sum R$. While ECCP keeps the cross traffic $\sum R$ less than maximum link capacity ($\sum R < C$), then the probe overhead never exceeds 5% of link capacity ($X_{probes} < 0.5 \times AvT \times C$). Therefore the probe overhead depends on the link capacity not on the number of flows.

2.4.1.2 ECCP probe receiver

The ***probe receiver*** captures probe frames, retrieves probe information, and adds receiving time for each probe frame. Then, ECCP ***probe receiver*** forwards each probe train information to ECCP ***bandwidth estimator*** for additional processing.

2.4.1.3 ECCP bandwidth estimator

The ***bandwidth estimator*** estimates *AvBw* using a modified version of BART which is based on a self-induced congestion model.

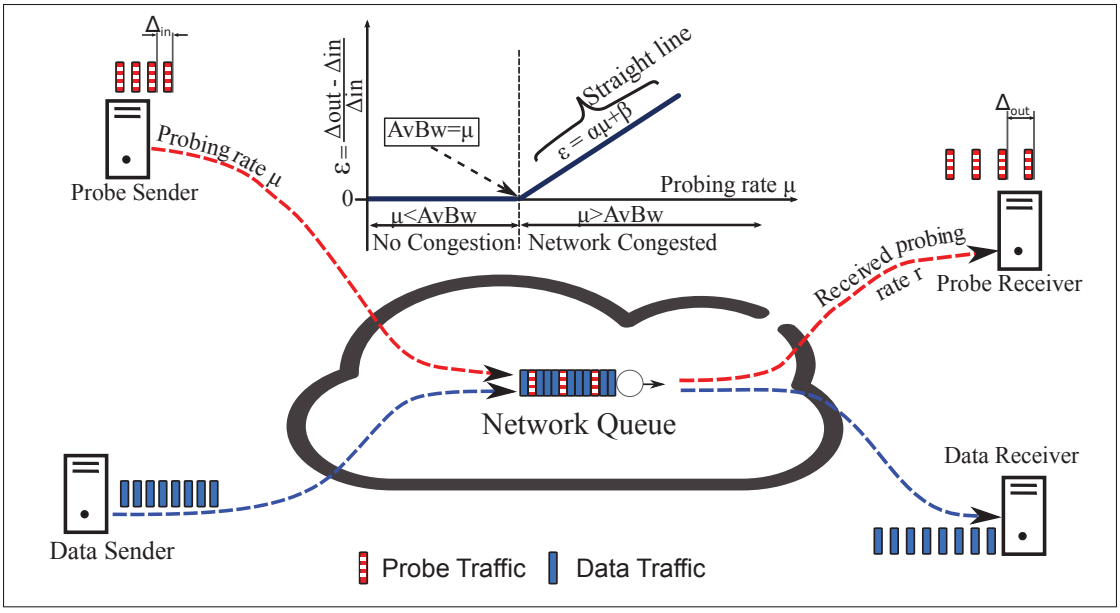


Figure 2.5 The effect of injecting probe traffic into network (Ekelin *et al.*, 2006)

In this model, when probe traffic of rate μ and fixed inter-frame intervals Δ_{in} is inserted along a network path, the received inter-frame interval Δ_{out} is affected by the network state such that; if μ is greater than *AvBw*, network queues start accumulating data which increases Δ_{out}. Otherwise, Δ_{out} will be, in average, equal to Δ_{in} (Fig. 2.5). This model does not require

clock synchronization between hosts. Rather, it uses the relative queuing delay between probe frames.

BART derives a new metric to define the change of the inter-frame time called strain $\varepsilon = (\Delta_{out} - \Delta_{in})/\Delta_{in}$. For probe rate μ that is less than *AvBw*, the strain will be, on average, equal to zero ($\varepsilon \approx 0$). Otherwise, the strain increases proportionally to the probe rate μ (Fig. 2.5). This linear relation between strain ε and probe rate μ is represented using (2.2).

$$\varepsilon = \begin{cases} 0 & \text{if } \mu \leq AvBw \\ \alpha\ \mu + \beta & \text{if } \mu > AvBw. \end{cases} \tag{2.2}$$

Based on this linear relationship between strain ε and probe rate μ, the ***bandwidth estimator*** calculates the strain ε_i for each probe pair $\{i = 1, \ldots, N-1\}$. Then, the calculated average ε and its variance R are forwarded to Kalman Filter (KF). In addition, an estimation of the system noise covariance Q and measurement error P are provided. Thus, KF estimates μ and β variables of the linear equation (2.2). Hence, ECCP estimates *AvBw* as the maximum probe rate that keeps the strain ε equal to zero $(\alpha \times AvBw + \beta = 0)$ as in (2.3).

$$AvBw = -\beta/\alpha. \tag{2.3}$$

For that purpose,

Kalman filter works on continuous linear systems while this model has a discontinuity separating two linear segments as shown in Fig. 2.5. Thus, BART ignores the probe rates μ that are not on the horizontal line where μ is less than the last estimated *AvBw* ($\mu < AvBw$). Unlike BART, ECCP does not ignore probe train information that is not on the straight line. Instead, it uses that probe rate μ to provide an estimation of *AvBw* using (2.4) (for more details see

(Bahnasy *et al.*, 2015)).

$$AvBw = \begin{cases} \max(\mu_j) & \text{if } \varepsilon < \varepsilon_t \\ \mathrm{KF}(\varepsilon_t, Q, P) & \text{if } \varepsilon \geq \varepsilon_t \end{cases} \tag{2.4}$$

where j is the probe train number and ε_t is the strain threshold that identifies the starting point of the straight line. Afterward, ***bandwidth estimator*** sends estimated *AvBw* back to the source in a CNM message.

2.4.1.4 ECCP rate controller

ECCP ***rate controller*** is a key component of ECCP mechanism. It controls the transmission rate R according to Additive Increase Multiplicative Decrease (AIMD) model after receiving *AvBw* value. Based on the received estimated *AvBw*, ECCP ***rate controller*** calculates available bandwidth ratio A_r according to (2.5). A_r represents the ratio of the available bandwidth to the bandwidth stability margin $(R \times AvT)$ (Fig. 2.6).

$$A_r = \frac{AvBw}{R \times AvT}. \tag{2.5}$$

ECCP works on keeping A_r at an equilibrium level A_{eq}. Therefore, it calculates a feedback parameter Fb to represent the severity of the congestion using (2.6).

$$Fb = (A_r - A_{eq}) + w \times (A_r - A_{old}) \tag{2.6}$$

where A_{old} is the previous value of A_r, and w is equal to 2 (similar to QCN) and it represents a weight for $(A_r - A_{old})$; i.e., w makes calculated Fb more sensitive to flows that change their rate aggressively than flows with stable high rates. Consequently, ECCP uses the calculated Fb to control hosts' transmission rate.

Furthermore, ECCP ***rate controller*** monitors two variables (1) the transmission rate R and (2) the target rate TR. TR is the transmission rate before congestion and represents an objective

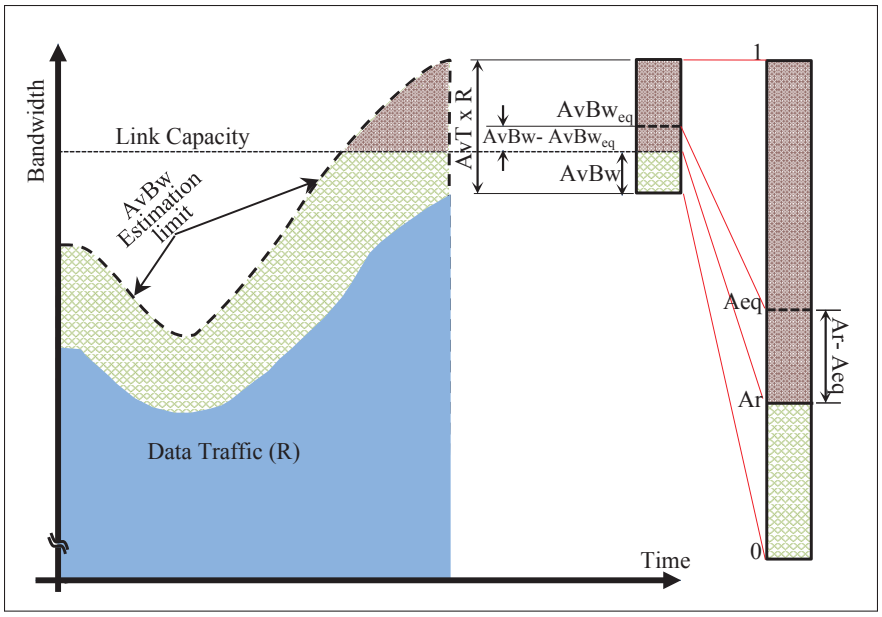


Figure 2.6 Relationship between $AvBw$ and A_r

rate for the rate increase process. ECCP ***rate controller*** uses a rate decrease process if the calculated Fb value is negative otherwise it uses a self-increase process as depicted by (2.7) (Fig. 2.7).

$$\begin{cases} \begin{cases} TR \leftarrow R \\ R \leftarrow R(1 + G_d \times F_b) \end{cases} & \text{if } F_b < 0 \text{ (rate decrease process)} \\ R \leftarrow \frac{1}{2}(R + TR) & \text{otherwise (Self-increase process)} \end{cases} \tag{2.7}$$

where G_d is a fixed value and is taken to make the maximum rate reduction equal to $1/2$.

Fig. 2.7 shows the ECCP rate control process in detail. The figure shows that when ECCP calculates a negative Fb, it executes the rate decrease process. In addition, Fig. 2.7 depicts that ECCP divides the self-increase process into three stages i) Fast Recovery (FR), ii) Active Increase (AI) and iii) Hyper-Active Increase (HAI). ECCP determines the increasing stage based on a byte counter BC and a timer T. The Fast Recovery stage consists of five cycles where each cycle is defined by sending BC Bytes of data or the expiration of a timer T. The

Algorithm 1: ECCP rate decrease process

Input : Available Bandwidth $AvBw$

Output: New sending rate R

$A_r \leftarrow AvBw/R \times AvT$;

$F_b \leftarrow (A_r - A_{eq}) + w \times (A_r - A_{old})$;

if $(Fb < 0)$ **then**

    $TR \leftarrow R$;

    $R \leftarrow R(1 + G_d \times F_b)$; /* Rate decrease */

    $SelfIncreaseStarted \leftarrow TRUE$; /* Initialize self-increase */

    $ByteCycleCnt \leftarrow 0$;

    $TimeCycleCnt \leftarrow 0$;

    $ByteCnt \leftarrow BC$;

    $Timer \leftarrow T$;

end

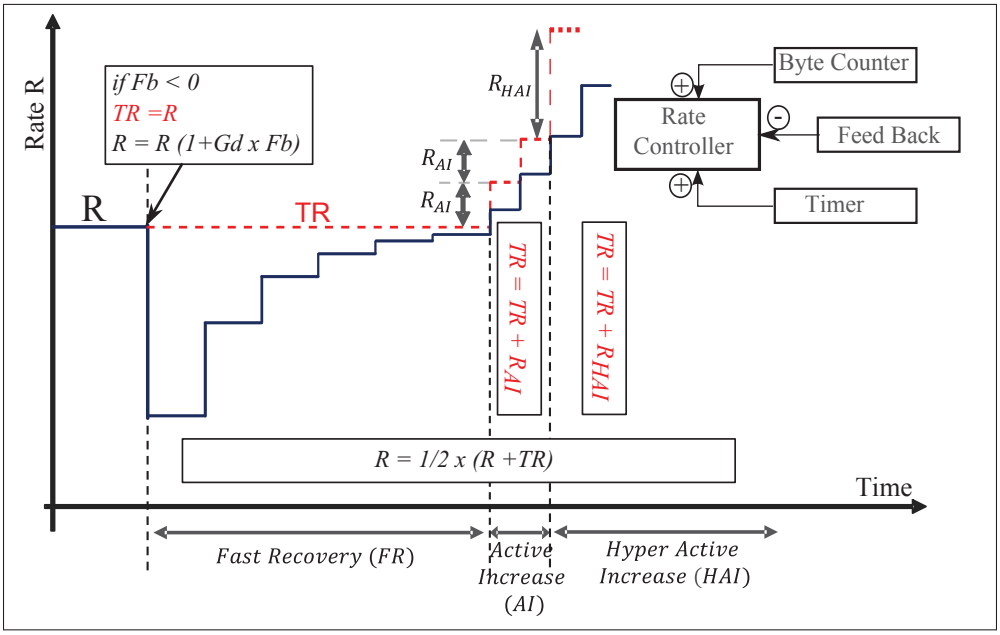


Figure 2.7 ECCP rate control stages

timer defines the end of cycles in case of low rate flows. At each cycle, R is updated using (2.7) while keeping TR unchanged. If the byte counter or the timer completes five cycles in FR stage while no negative Fb is calculated, the ***rate controller*** enters Active Increase (AI)

Algorithm 2: ECCP self-increase process

```
1  Output: New sending rate R
2  foreach sentFrame do
3      if SelfIncreaseStarted == TRUE then
4          ByteCnt ← ByteCnt − Byte(frameSize) ;
5          if (ByteCnt ≤ 0) then
6              ByteCycleCnt++ ;
7              if (ByteCycleCnt < 5) then
8                  ByteCnt ← BC ;
9              else
10                 ByteCnt ← BC/2 ;
11             AdjustRate() ;
12 end
13 foreach timeout do
14     if SelfIncreaseStarted == TRUE then
15         TimeCycleCnt++ ;
16         if (ByteCycleCnt > 5) or (TimeCycleCnt > 5) then
17             RestartTimer(T/2) ;                    /* AI or HAI stage */
18         else
19             RestartTimer(T) ;                      /* FR stage */
20         AdjustRate() ;
21 end
22 ---------------------------------------------------------------
23 AdjustRate():
24 if (ByteCycleCnt > 5) and (TimeCycleCnt > 5) then
25     TR ← TR + 5 Mbps ;                             /* HAI stage */
26 else if (ByteCycleCnt > 5) or (TimeCycleCnt > 5) then
27     TR ← TR + 50 Mbps ;                            /* AI stage */
28 R ← 1/2 × (R + TR);
29 if (R > linkCapacity) then
30     R ← linkCapacity; SelfIncreaseStarted ← FALSE;
```

stage. In this stage, TR is increased by a predefined value R_{AI}. Moreover, the byte counter and the timer limits are set to $BC/2$ and $T/2$ respectively. Afterward, the ***rate controller*** enters the Hyper-Active Increase (HAI) stage, if both the byte counter and the timer finish five cycles. In

the HAI stage, TR is increased by a predefined value R_{HAI} as in (2.8).

$$TR \leftarrow \begin{cases} TR + R_{AI} & (\text{AI}) \\ TR + R_{HAI} & (\text{HAI}). \end{cases} \tag{2.8}$$

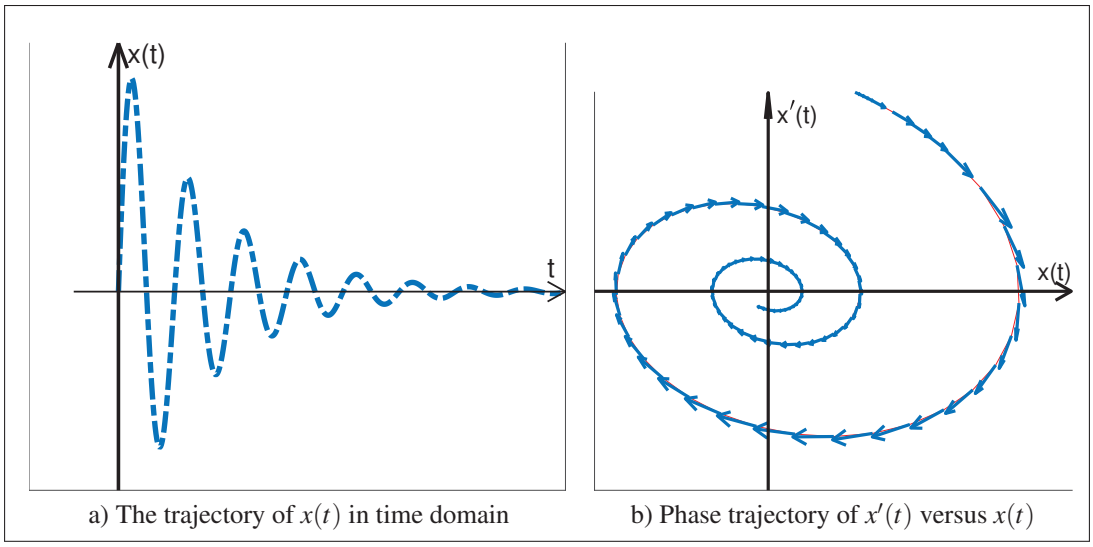
Where R_{AI} is the rate increase step in AI stage and R_{HAI} is the rate increase step in HAI stage. Algorithms 1 and 2 depict ECCP rate decrease and self-increase processes respectively.

2.5 Phase Plane Analysis

In this paper, we use phase plane method to visually represent certain characteristics of the differential equation of the ECCP. Phase plane is used to analyze the behavior of nonlinear systems. The solutions of differential equations are a set of functions which could be plotted graphically in the phase plane as a two-dimensional vector field. Given an autonomous system represented by a differential equation $x''(t) = f(x(t), x'(t))$, one can plot the phase trajectory of such a system by following the direction where time increases. Fig. 2.8a depicts a system $x(t)$ in time domain, where a phase trajectory of this system is displayed in Fig. 2.8b. One can notice that $x(t)$ and $x'(t)$ in time domain can be inferred from the phase trajectory plot. Thus, the phase trajectory provides enough information about the behavior of the system. Moreover, sketching phase trajectories is easier than finding an analytical solution of differential equations, which sometimes is not possible.

Congestion control schemes in computer networks require different behaviors for rate increase and rate decrease subsystems. In addition, the congestion state controls the transition between these subsystems. The phase plane method could link isolated subsystems and present graphically the switching process. Thus, using phase plane method is adequate for analyzing segmented systems like congestion control protocols (Jiang *et al.*, 2015).

In addition, system parameters limitation can be taken into consideration explicitly. Therefore, we should consider only the phase trajectories that satisfy our system limitations (i.e link capacity and buffer size) even if the system is stable according to the derived stability conditions.



a) The trajectory of $x(t)$ in time domain

b) Phase trajectory of $x'(t)$ versus $x(t)$

Figure 2.8 Phase trajectory example

2.6 ECCP Modeling

The core element of ECCP is the rate control algorithm. By responding correctly to the calculated feedback, the network load should remain around the target point. For the purpose of simplicity, we made these assumptions:

- All sources are homogeneous, namely they have the same characteristics such as round-trip time.
- Data flows in data center networks have high rates and appear like continuous flow fluid.
- Available bandwidth estimation error is negligible (Measured *AvBw* is used in the simulation to avoid estimation errors). We leave studying the effect of available bandwidth estimation error on ECCP stability for future work.

Given the aforementioned assumptions, ECCP can be modeled while calculating the available bandwidth using (2.9).

$$AvBw(t) = C - M \times R(t) \tag{2.9}$$

where $AvBw(t)$ is the available bandwidth at time t, C is the maximum link capacity, M is the number of flows that share the same bottleneck link, and $R(t)$ is the host's transmission rate at time t.

By substituting (2.9) into (2.5) we get:

$$A_r(t) = \frac{C}{AvT \times R(t)} - \frac{M}{AvT} \tag{2.10}$$

where $A_r(t)$ is the available bandwidth ratio at time t.

In addition, feedback calculation in (2.6) becomes:

$$Fb(t) = A_r(t) - A_{eq} + w \times T \times A'_r(t-T) \tag{2.11}$$

where T is the time interval between trains which defines the control cycle time, and $(A_r - A_{old})$ becomes the derivative of availability ratio A'_r multiplied by the control cycle time T.

Given ECCP rate update equation (2.7), the derivative of transmission rate $R'(t)$ can be represented by the delay differential equation (2.12).

$$R'(t) = \begin{cases} \frac{G_d}{T} R(t) Fb(t-\tau) & \text{if } Fb(t-\tau) < 0 \\ \frac{TR - R(t)}{2 \times T_{BC}} & \text{if } Fb(t-\tau) \geq 0 \end{cases} \tag{2.12}$$

where τ is the propagation delay, T_{BC} is the BC counter time.

2.7 Stability Analysis of ECCP

In this section, phase plane is used in studying the stability of ECCP. Phase plane analysis of ECCP is carried out for the self-increase and rate decrease processes separately. Next, simulation experiments are presented to verify our mathematical analysis.

2.7.1 Stability Analysis of ECCP Rate Decrease Subsystem

In this section, we analyze ECCP rate decrease subsystem represented by (2.12), (2.11), and (2.10). For the sake of simplicity and without loss of generality, we made this linear variable

substitution.

$$\begin{cases} y(t) & = A_r(t) - Aeq \\ y'(t) & = A'_r(t). \end{cases} \tag{2.13}$$

Thus, from (2.10) we get:

$$y(t) = \frac{C}{AvT \times R(t)} - \frac{M}{AvT} - A_{eq}$$

Let $\zeta = \frac{M}{AvT} + A_{eq}$, we get:

$$\begin{aligned} y(t) &= \frac{C}{AvT \times R(t)} - \zeta \\ R(t) &= \frac{C}{AvT \times (y(t) + \zeta)} \\ R'(t) &= -\frac{C}{AvT \times (y(t) + \zeta)^2} y'(t). \end{aligned} \tag{2.14}$$

The feedback equation could be represented by substituting (2.13) in (2.11):

$$Fb(t) = y(t) + w \times Ty'(t - T). \tag{2.15}$$

Substituting (2.14) and (2.15) into the rate decrease part of (2.12), we get the rate decrease subsystem equation (2.16).

$$\begin{aligned} \frac{-C}{AvT(y(t) + \zeta)^2} y'(t) &= \frac{G_d}{T} Fb(t - \tau)\left(\frac{C}{AvT(y(t) + \zeta)}\right) \\ \frac{-1}{(y(t) + \zeta)} y'(t) &= \frac{G_d}{T} Fb(t - \tau) \\ -y'(t) &= \frac{G_d}{T} Fb(t - \tau)(y(t) + \zeta). \end{aligned} \tag{2.16}$$

Thus, ECCP rate decrease subsystem could be represented by substituting (2.15) into (2.16).

$$-y'(t) = \frac{G_d}{T}\Big(y(t-\tau) + w \times T \times y'(t-T-\tau)\Big)(y(t)+\zeta). \tag{2.17}$$

Based on (2.17), we can state this lemma.

Lemma 2.1. *ECCP rate decrease subsystem is stable when (2.18) is satisfied.*

$$\tau/T < min\Big(w - \frac{1}{G_d\zeta} + \sqrt{2w^2 - \frac{2}{(G_d\zeta)^2} + 4w},\ w + \frac{1}{G_d\zeta},\ w + \sqrt{w^2+2w}\Big). \tag{2.18}$$

For proof, review Appendix I.

2.7.2 Stability Analysis of ECCP Rate Increase Subsystem

The self-increase subsystem behavior can be summarized as follows: The stability of ECCP system mainly depends on the sliding mode motion (Utkin, 1977) from self-increase subsystem into the rate decrease subsystem when the system crosses the asymptotic line ($Fb = 0$). Thus, the ECCP system is asymptotically stable when inequality (2.18) is satisfied. For proof, review Appendix II.

2.7.3 Verification of ECCP's stability conditions using simulation

In this section, we use discrete-event simulation to verify the mathematical analysis of ECCP. Using OMNEST network simulation framework (Varga & Hornig, 2008), we simulate a dumbbell topology of four data sources and four receivers connected to two 10-Gbps switches as shown in Fig. 2.9. All links in this topology have a maximum capacity of 10 Gbps. We consider the worst case which happens when all sources send with their maximum link capacity. Thus, we have four data sources that send data at maximum line capacity (10 Gbps) toward four receivers through one bottleneck link (Fig. 2.9). Table 2.1 depicts the simulation parameters.

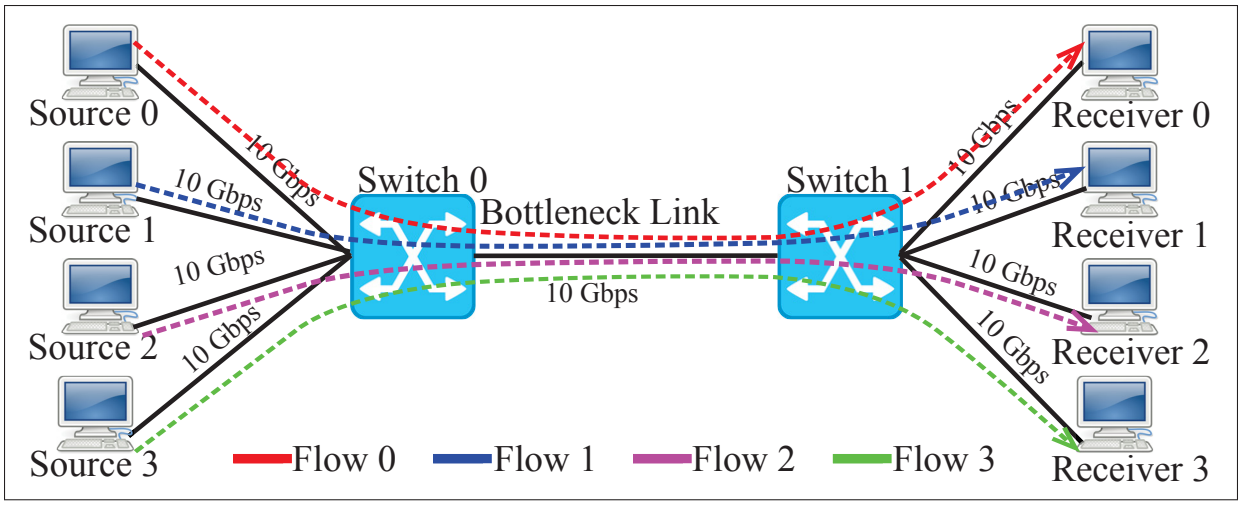


Figure 2.9 Simulation topology

Based on ECCP parameters that are shown in Table 2.1 and inequality (2.18), ECCP is stable for all $\tau < 1.482\ T$. Fig. 2.10 shows a box plot of cross traffic. It depicts that ECCP system

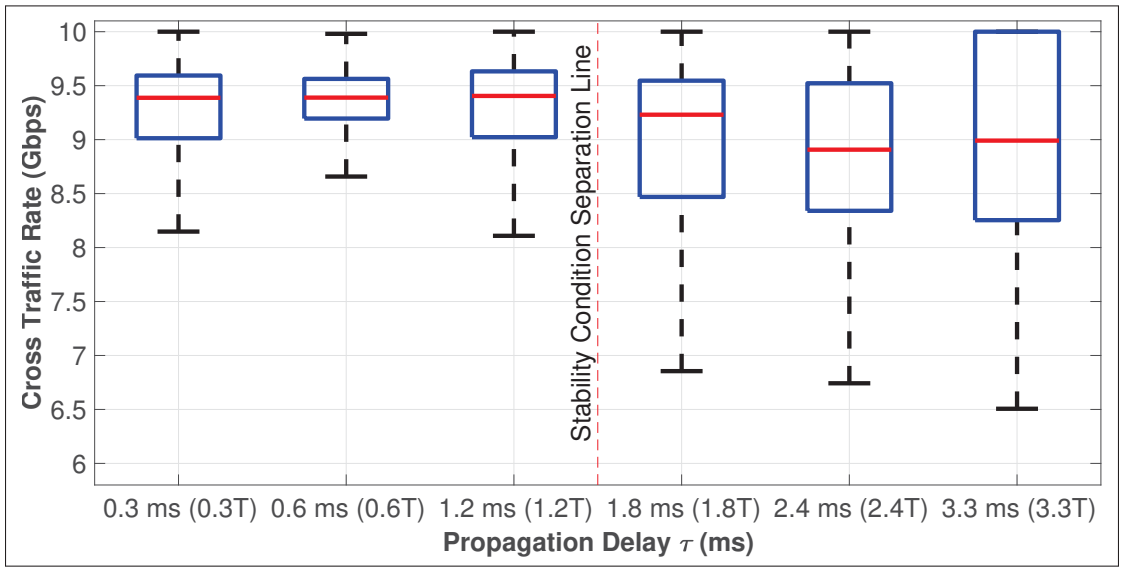


Figure 2.10 Box plot of the cross traffic rate

reduces cross traffic rate to a value lower than its minimum limit ($(1 - AvT) \times C = 9$ Gbps), when τ exceeds the analytically calculated limit $(1.482\ T)$. In addition, Fig. 2.10 clearly shows that when $\tau = 1.8\,ms > 1.482\ T$, the variation of cross traffic exceeds the maximum allowed

Table 2.1 Simulation parameters

Data Senders Parameters	
Frame size	Normal distribution ($avg = 1000, \sigma = 150$)
Min Frame size	200 Byte
Max Frame size	1500 Byte
Propagation Delay τ	$40\,\mu sec$
ECCP Probing Parameters	
Number of probe frames	$N = 33$
Size of probe frames	1500 Byte
Time between each train	1 ms
Available Threshold	$AvT = 0.1$ (10%)
Minimum probe rate	50 Mbps
Maximum probe rate	$AvT \times R$
System noise	$Q = \begin{pmatrix} 0.00001 & 0.0 \\ 0.0 & 0.01 \end{pmatrix}$
Measurement error	$P = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 100.0 \end{pmatrix}$
ECCP Controller Parameters	
Equilibrium available bandwidth ratio	$A_{eq} = 0.5$ (50%)
Rate control timer	T = 5 ms
Rate control byte counter	BC = 750 KByte
G_d	G_d = 100/128
Rate increase step in AI stage	R_{AI} = 5 Mbps
Rate increase step in HAI stage	R_{HAI} = 50 Mbps

margin ($AvT \times C = 1$ Gbps). One can notice that when $\tau = 3.3\ T$ the average cross traffic starts to increase again. The reason behind that is the data accumulation in the queue as shown in Fig. 2.11b.

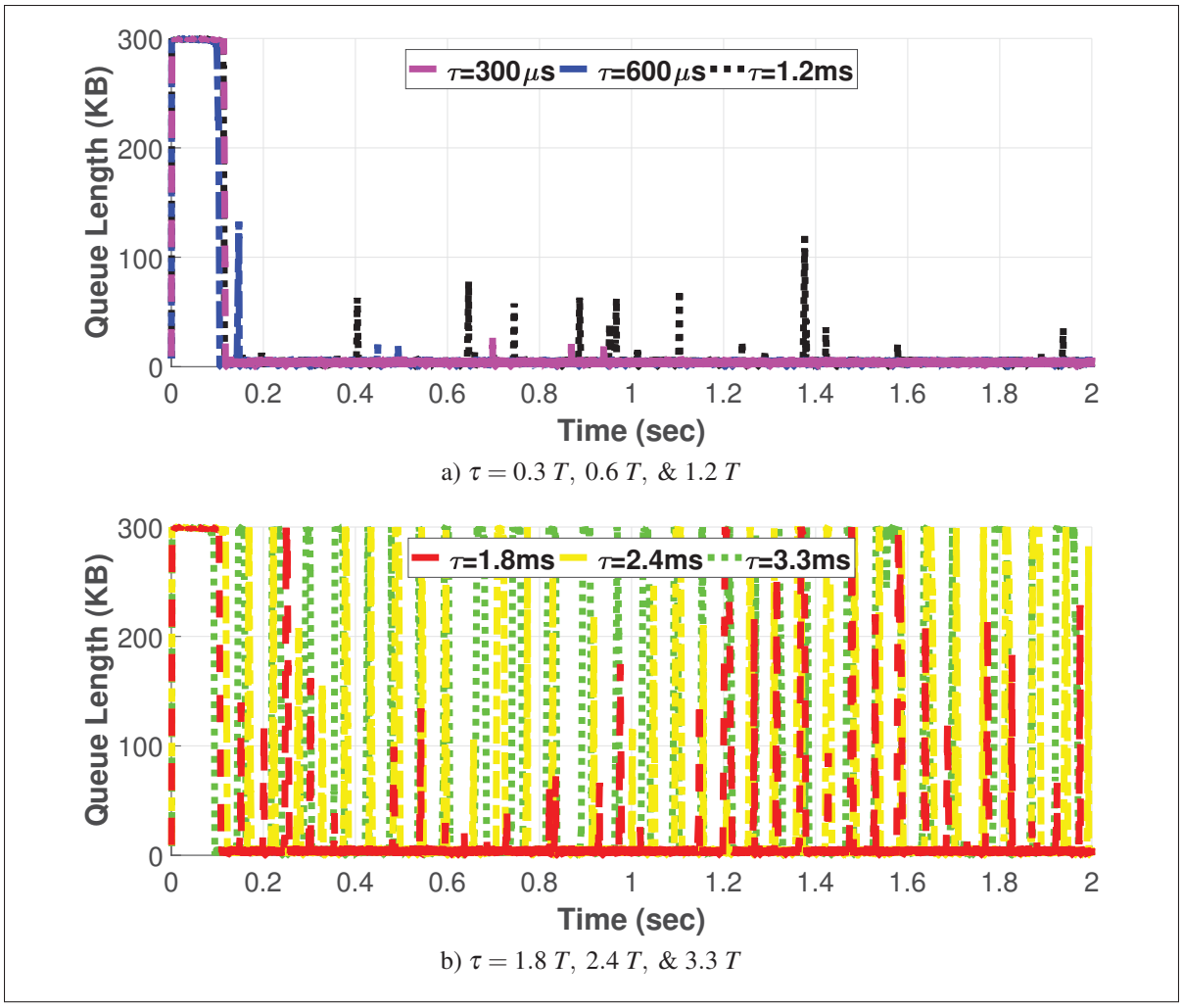


Figure 2.11 Queue length

Fig. 2.11 depicts the queue length while varying the propagation delay ($\tau = 0.3\ T$, $0.6\ T$, $1.2\ T$, $1.8\ T$, $2.4\ T$ & $3.3\ T$). Fig. 2.11a shows that if the stability conditions are satisfied ($\tau < 1.482T$), ECCP system succeeds in maintaining a close-to-zero queue length. Otherwise, data start to accumulate and the queue fluctuates significantly as shown in Fig. 2.11b.

Fig. 2.12 illustrates the cumulative distribution function (CDF) of the queue length. It shows that when stability conditions are satisfied and $\tau = 0.3\ T$, $0.6\ T$, $1.2\ T$, 99-percentile of queue length are less than 6.72 KB, 6.78 KB and 21.9 KB respectively. But when these conditions are violated, 99-percentile of queue length reach up to 294.4 KB.

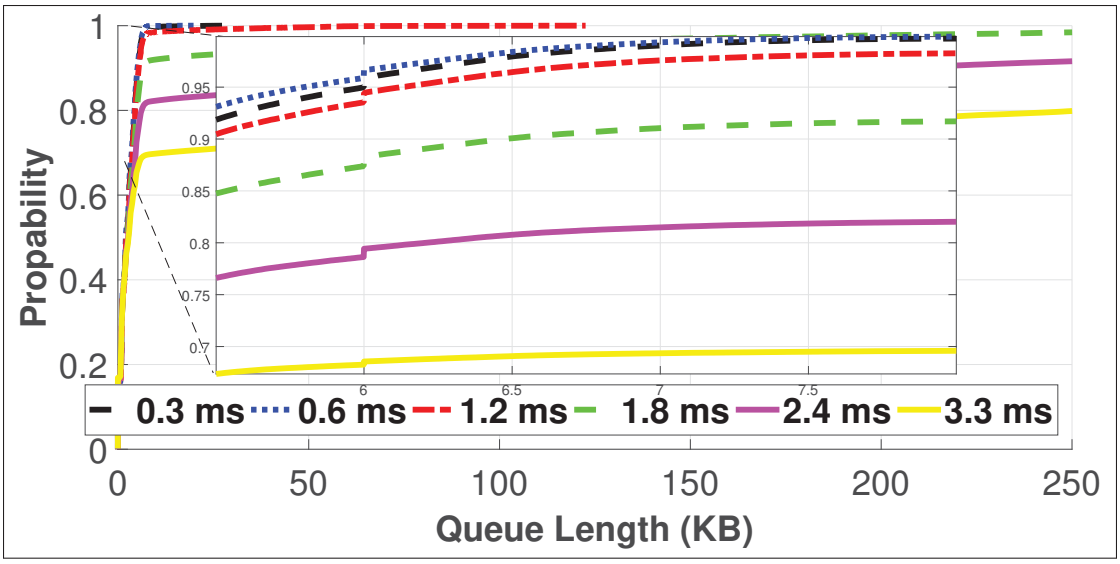


Figure 2.12 CDF of queue length

Fig. 2.13 depicts the transmission rates while varying the propagation delay. It shows that as long as τ does not exceed the stability limit (1.482 T), ECCP system achieves fairness between flows.

2.7.4 Boundary limitations

In our stability analysis, we have deduced sufficient stability conditions of the core mechanism of ECCP. However, ECCP system is also constrained by physical boundaries such as the maximum link capacity and buffer size. For example, when the ECCP system reaches the equilibrium point, hosts keep increasing their data rates until calculating a positive Fb. Thus, cross traffic might reach the maximum limit and data starts to be queued in the system. In order to avoid this, the integral of the self-increase function from t to $(t+(T+2\tau))$ must be less than the available bandwidth margin ($AvT \times A_{eq} \times C$), where $(T+2\tau)$ is the control cycle time. The boundary limitation of ECCP queue system is summarized by the following lemma.

Lemma 2.2. *ECCP keeps queue length close to zero, thereby ensuring minimum network latency and preventing congestion if inequality (2.19) is satisfied.*

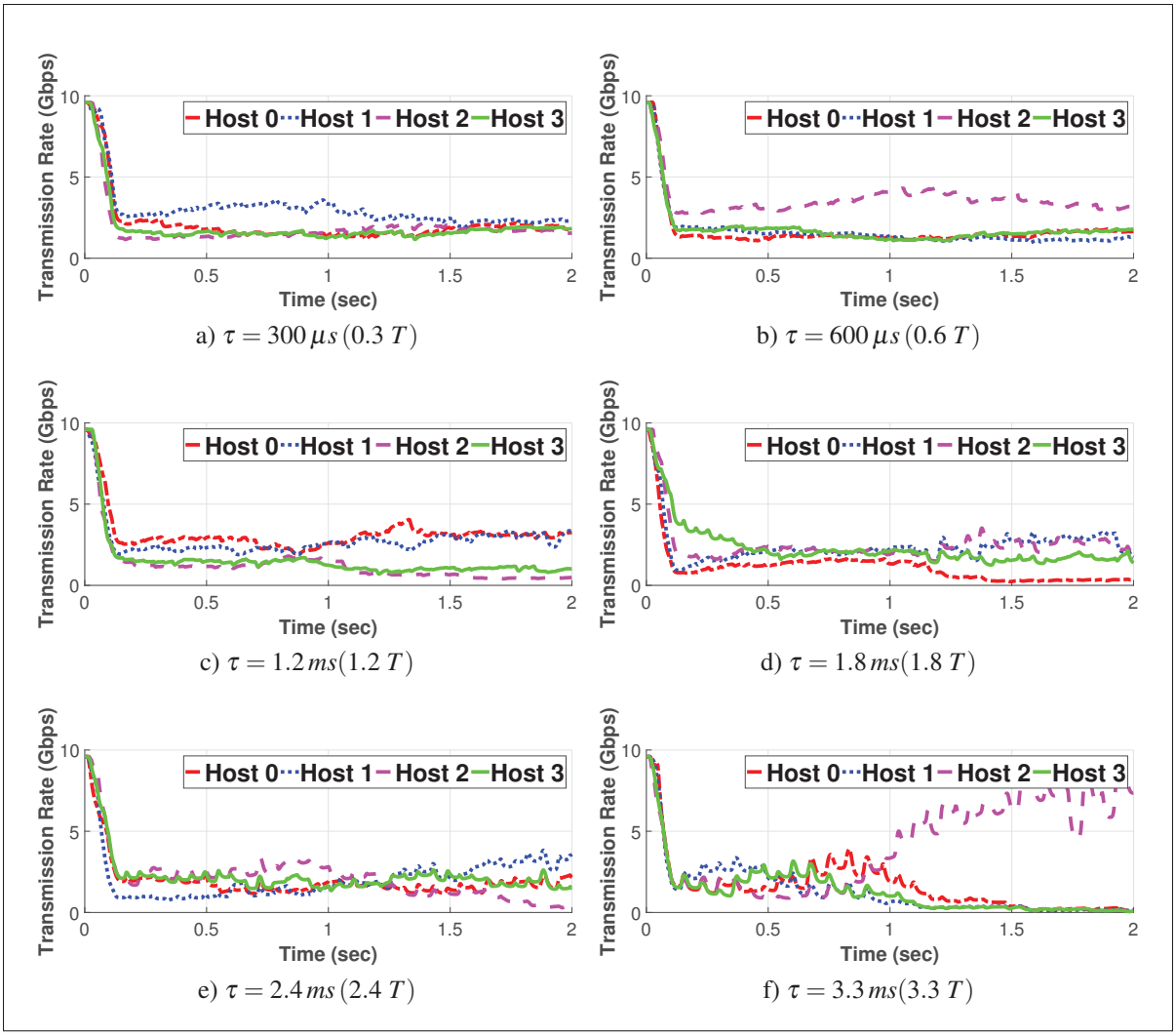


Figure 2.13 Transmission rates

$$BC > \frac{C(T+2\tau)}{2M}. \tag{2.19}$$

The proof of lemma 2.2 is presented in Appendix III.

2.7.5 Verification of ECCP's boundary limitations using simulation

Based on ECCP parameters shown in table 2.1 and inequality (2.19), ECCP is capable of keeping queue length close to zero when $BC > 500\,KB$. ECCP is simulated to verify the analytical model. Fig. 2.14, 2.15, 2.16 and 2.17 depict the simulation results while varying the byte counter (BC = 150 KB, 450 KB, 600 KB, and 750 KB). In addition, Fig. 2.14 shows that

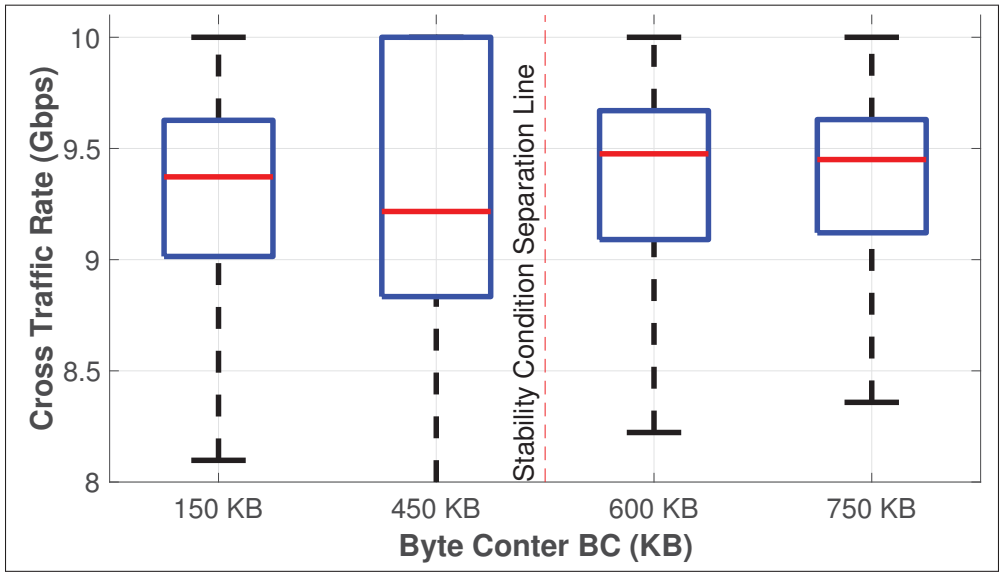


Figure 2.14 Cross traffic statistics

when inequality (2.19) is not satisfied ($BC < 500\,KB$), ECCP system becomes unstable and the cross traffic variation exceeds ($AvT \times C$) limit (1 Gbps). It is clearly shown that reducing BC decreases the average cross traffic rate and increases its variation. One can notice that at $BC = 150KB$, the average cross traffic rate starts to increase again which is a result of data accumulation in the bottleneck link queue as shown in Fig. 2.15. Besides, Fig. 2.15 depicts that when byte counter does not satisfy the analytically calculated limit $BC < 500\,KB$, the queue starts accumulating data. In contrast, when byte counter limit is satisfied $BC > 500\,KB$, ECCP succeeded in maintaining a close-to-zero queue length.

Fig. 2.16 shows the CDF of the queue length. It depicts that when BC is equal to 750 and 600 KB, 99-percentile of queue length are less than 6.9 KB and 6.8 KB respectively. But when

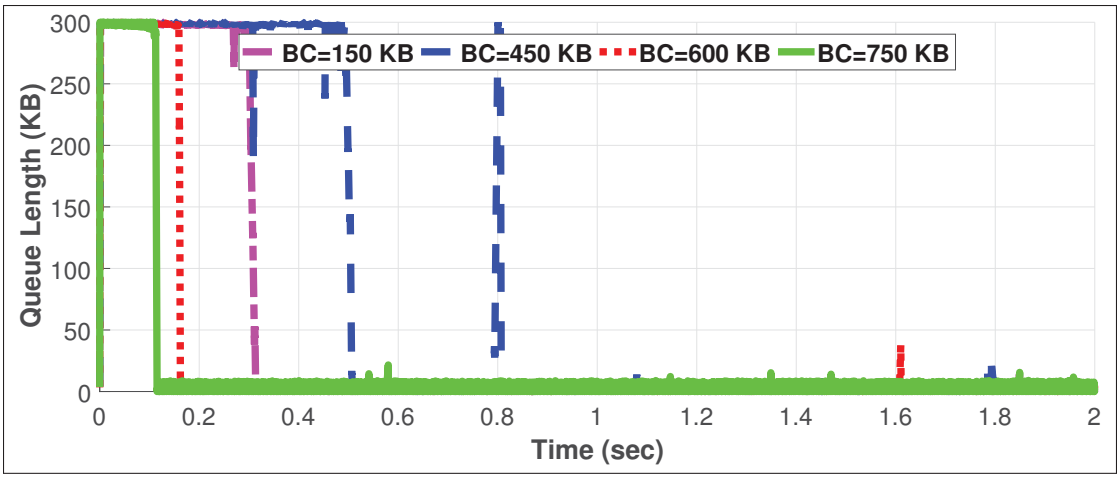


Figure 2.15 Queue length

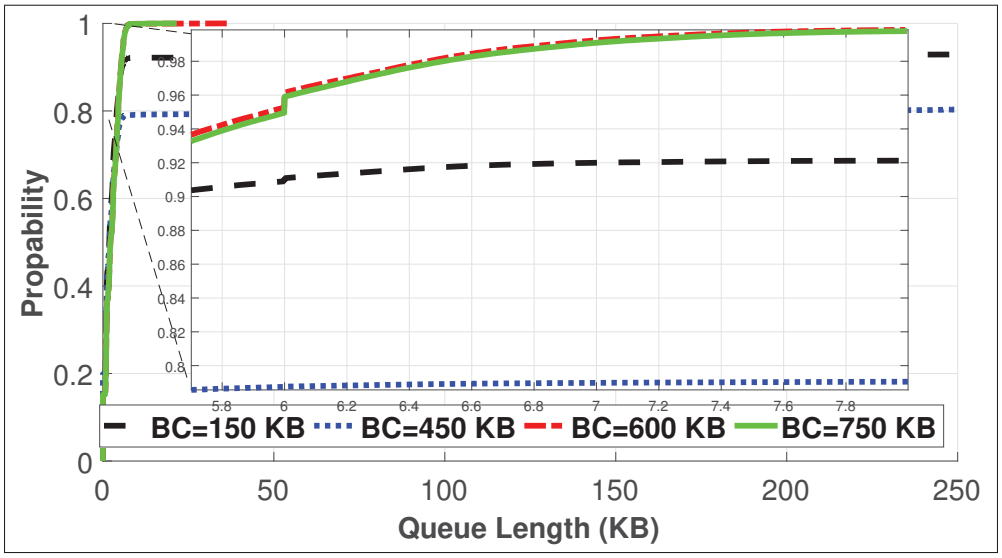


Figure 2.16 CDF of queue length

inequality (2.19) is not satisfied where $BC < 500$ KB, 99-percentile of queue length reaches up to 299 KB.

Fig. 2.17 depicts the effect of varying the byte counter BC on the transmission rates. It shows that when $BC = 150$ and $450\,KB$, flows with high rate start recovering faster than flows with low rate (Fig. 2.17a & 2.17b) but when $BC = 600$ and $750\,KB$, hosts start to recover at a relatively equal speed which achieves fairness between flows (Fig. 2.17c & 2.17d). This limit

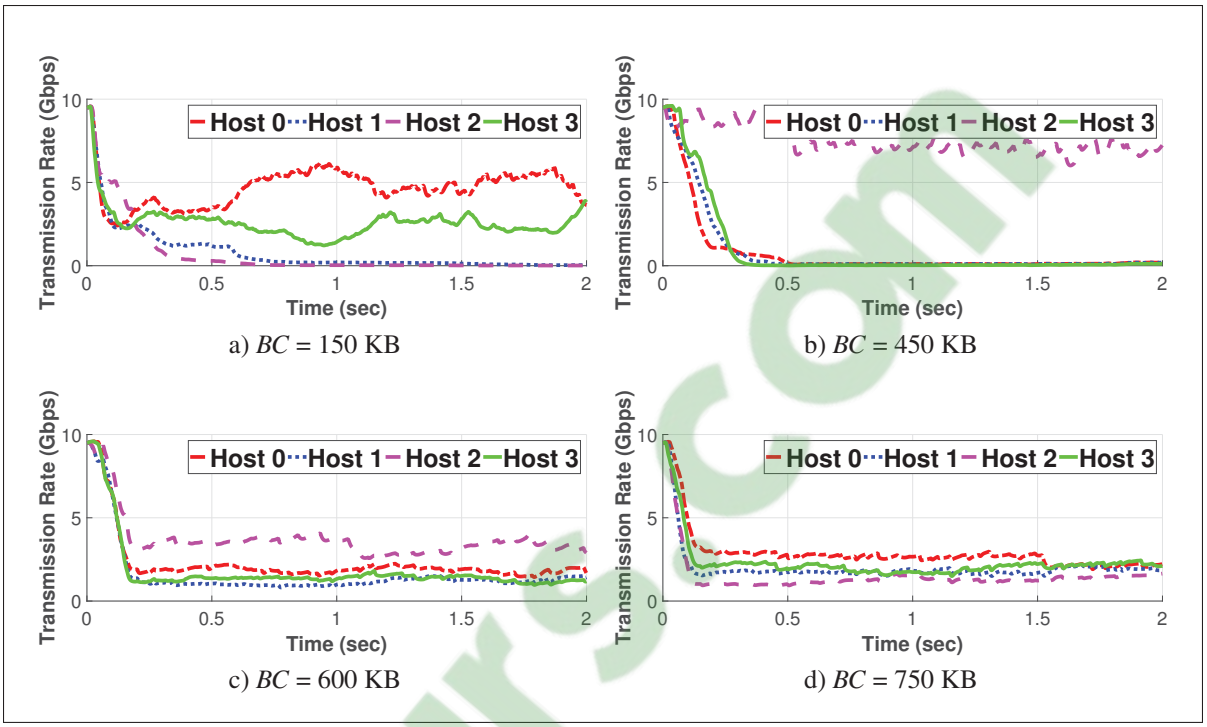


Figure 2.17 Transmission rates

matches the predicted value by the analytical analysis (Inequality 2.19). One can notice that when $BC = 750$, hosts' transmission rates were between 1.49 and 2.293 *Gbps*. In addition, when $BC = 600$, hosts' transmission rates were between 1.5928 and 3.7226 *Gbps* which still in the a fair range.

2.7.6 Discussion

The time interval between trains T must be greater than the sending time of the whole train (N frames, of 1500 Byte each) with rate equals to $AvT \times R_{min}$, where R_{min} is the minimum probing rate.

$$T \geq \frac{N \times 1500 \times 8}{AvT \times R_{min}}. \tag{2.20}$$

Furthermore, T determines the control cycle which controls the buffer boundary. For example, for a stable system of M number of flows, ECCP will keep the queue length close to zero. If a

new flow arrives with rate equals R_0, thus, R_0 must satisfy:

$$R_0 \times T \leq B. \tag{2.21}$$

Where B is the maximum switch buffer size. In other words, the hardware buffer inside the switch must satisfy $B \geq T \times R_0$, or any new flow has to start with rate $R_0 \leq \frac{B}{T}$.

2.8 Linux-Based Implementation

We have implemented an ECCP testbed using 3 Linux hosts and a 10 Gbps switch. In this implementation we used Linux Traffic Control (Hubert *et al.*, 2002) to separate probe traffic from data traffic and to throttle the transmission rate of data traffic as explained in Section 2.8.2. The testbed is connected as shown in Fig. 2.18 and is configured according to table 2.1. In this implementation, we built a Java GUI to periodically collect statistics and plot the actual transmission rate R, and cross traffic rate at the receiver (Fig. 2.21).

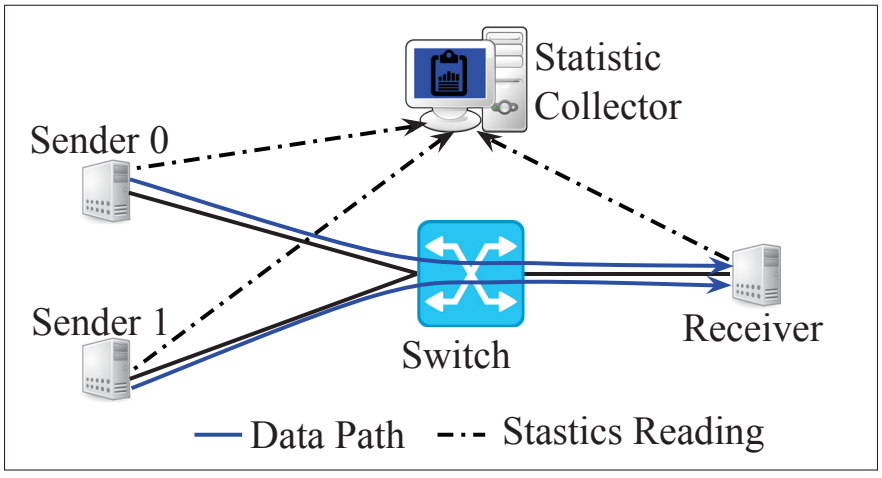


Figure 2.18 Experiment testbed topology

In the next section we present several experiments to validate our bandwidth estimation method, and in the following section we present the ECCP testbed implementation.

2.8.1 Validating available bandwidth estimation process

ECCP's available bandwidth estimation process is tested using the aforementioned testbed. In this topology, sender 0 sends a constant bit rate traffic to the receiver and sender 1 sends probe traffic with a randomly generated rate μ. Fig. 2.19 shows the measured strains ϵ versus the probe rate μ at the receiver in three scenarios (i) $AvBW = 6$ Gbps, (ii) $AvBW = 5$ Gbps, (iii) $AvBW = 1.5$ Gbps. Fig. 2.19 depicts that when ϵ starts increasing, μ is always identical to *AvBw* in all cases. Thus, we conclude that this method is trustworthy and could be used to estimate *AvBw*.

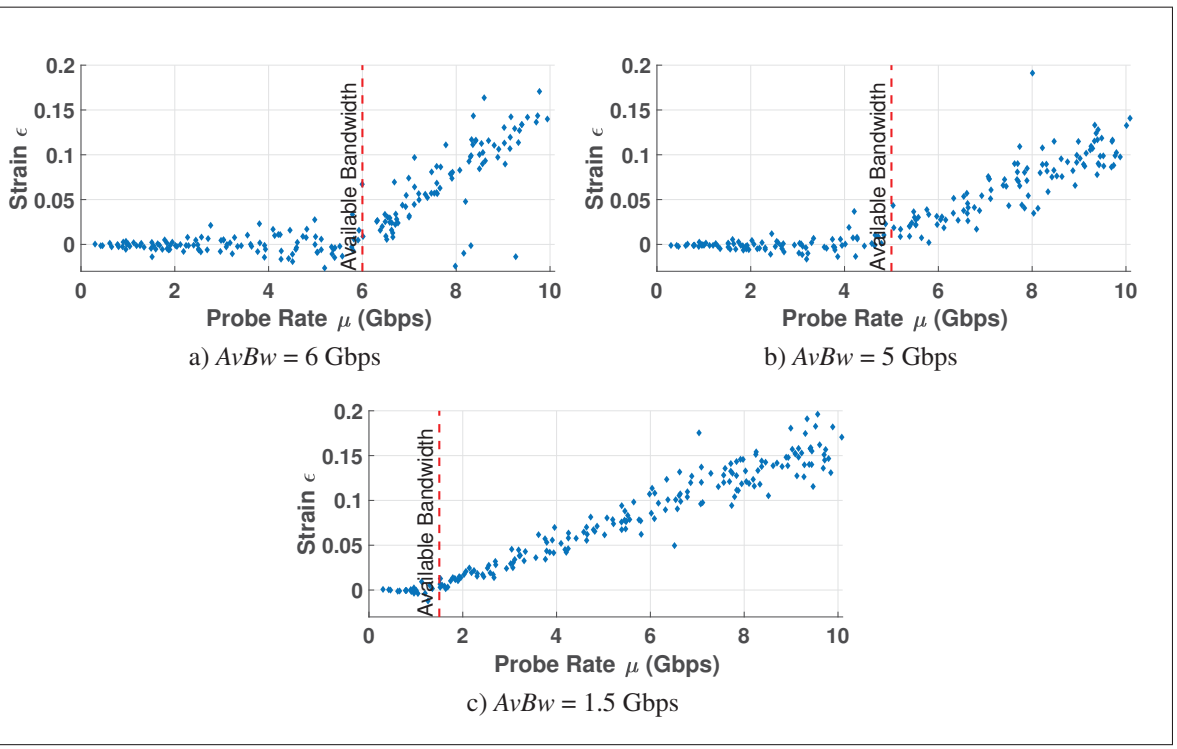
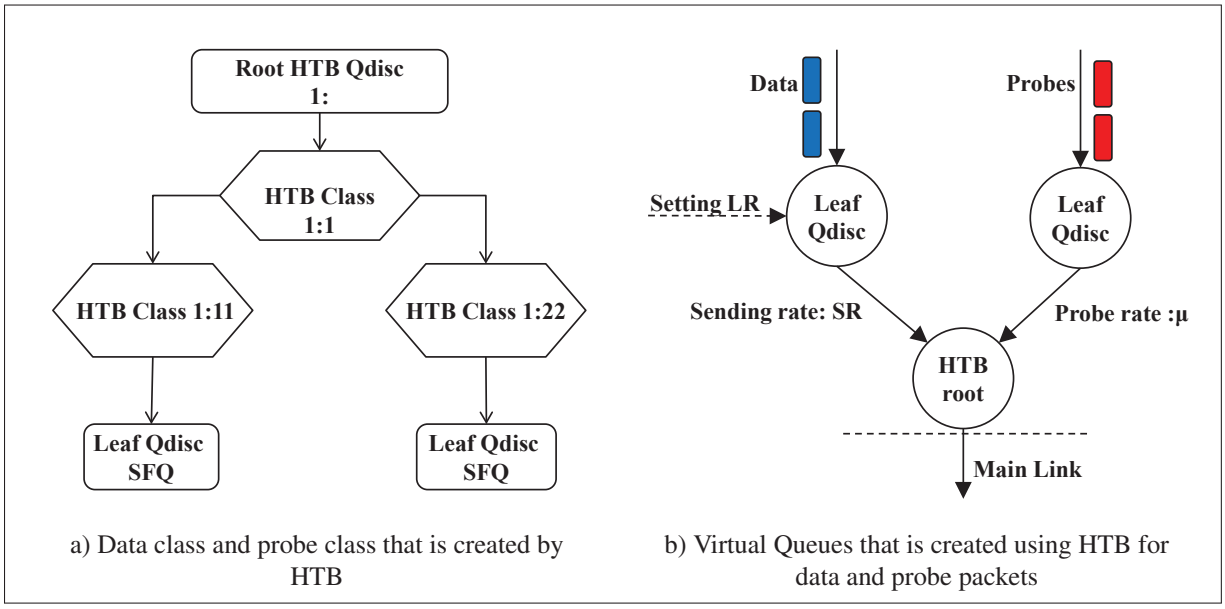


Figure 2.19 ECCP's available bandwidth estimation process

2.8.2 ECCP testbed implementation

In this testbed, we have implemented the ECCP ***rate controller*** using Linux Traffic Control (Hubert *et al.*, 2002). ECCP needs to control data traffic while probe traffic must be forwarded

with no control. To achieve this goal, we use Hierarchy Token Bucket (HTB) (Devera, 2002) to create two virtual schedulers (Qdisc) with different classes in Linux machines. HTB allows sending different classes of traffic on different simulated links using one physical link. HTB is used to ensure that the maximum service provided for each class is the minimum of the desired rate *DR* or the assigned rate *R* by ECCP. Fig. 2.20a shows the two classes that we create to represent data flow and probe flow. In addition, two virtual schedulers (Qdisc) are created and linked to these classes (Fig. 2.20b). Thus, ECCP can limit the data rate by setting the rate on class 1:11 equal the maximum allowed rate, while keeping the probe class (Class 1:22) uncontrolled. Note that these two queues have different priorities; data flow enter the queue with low priority while probe flow is forwarded through the queue with high priority.



a) Data class and probe class that is created by HTB

b) Virtual Queues that is created using HTB for data and probe packets

Figure 2.20 HTB virtual queues and their classes

In this experiment, each host sends with desired rate *DR* that are throttled by HTB to the sending rate *R* which is calculated by ECCP. *DRs* are varied 4 times in this test, in the first period ($0\ s < t < 4\ s$), host 0 sends with $DR = 4$ Gbps while host 1 sends with $DR = 1$ Gbps (Fig. 2.21). In this period, there is no congestion and the transmission rates *R* are not controlled (equal *DR*). In the second period ($4\ s < t < 12.4\ s$), host 1 increases its *DR* to 6 Gbps. Thus,

ECCP starts limiting *DR* by setting *R* to a value that keeps the cross traffic close to 9.5 Gbps. One can notice in this period, ECCP controls only the greedy flow (Host 1) while allowing Host 0 to send with its *DR*. In the third period ($12.4\ s < t < 14.2\ s$), host 0 increases its *DR* to 6 Gbps. Therefore, ECCP starts to control both hosts' rates severely to prevent congestion. Finally, when $t > 14.2\ s$, host 0 decreases its *DR* to 3 Gbps which ends the congestion. Thus, ECCP alleviates its control, and each host sends with its desired rate ($R = DR$).

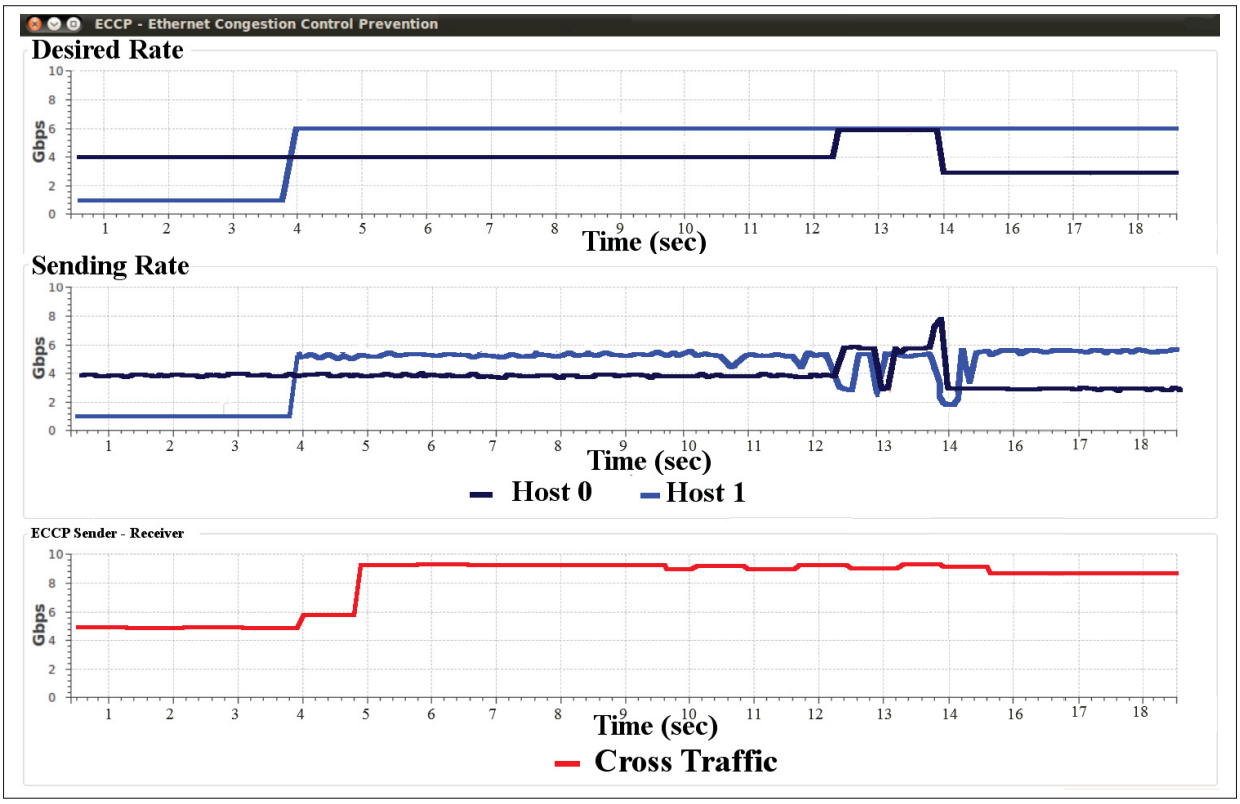


Figure 2.21 ECCP lab implementation results

2.9 Conclusion

In this paper, we propose ECCP as a distributed congestion control mechanism that is implemented in line cards or end hosts and does not require any switch modification.

We analyzed ECCP using phase plane method while taking into consideration the propagation delay. Our stability analysis identifies the sufficient conditions for ECCP system stability. In addition, this research shows that the stability of the ECCP system is ensured by the sliding mode motion. However, the stability of ECCP depends not only on its parameters but also on the network configurations.

Several simulations were driven to verify our ECCP stability analysis. The obtained numerical results reveal that the ECCP system is stable when the delay is bounded. Finally, a Linux-based testbed experimentation is conducted to evaluate ECCP performance.

As a perspective of this work, we are presently (i) studying the effect of available bandwidth estimation error on ECCP stability, (ii) evaluating ECCP in larger and various network topologies using our simulator.

CHAPTER 3

FAIR CONGESTION CONTROL PROTOCOL FOR DATA CENTER BRIDGING

Mahmoud Bahnasy[1], Halima Elbiaze[2]

[1] Département de Génie électrique, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Département d'informatique, Université du Québec à Montréal

This article was submitted at «IEEE/ACM Transactions on Networking » in December 2017.

3.1 Abstract

DCN brought a new era of data-intensive applications such as HPC, cloud computing, SAN and VXLAN which raise new challenges to network researchers. Such applications require minimum network latency, no packet loss and fairness between flows. Therefore, IEEE DCB Task Group presents several enhancements for Ethernet network to fulfill these requirements. In this context, we explore the possibility of controlling congestion in Ethernet layer to achieve those requirements. We propose HetFlow as a delay-based congestion control mechanism that controls congestion while achieving minimum queue length, minimum network latency, and high link utilization. HetFlow guarantees fairness between flows of different packet sizes and different RTTs. In addition, we present a mathematical model, stability analysis and scalability study of the proposed protocol.

3.2 Introduction

The emergence of DCN and its applications raise the need for stable, robust and fair transport network in data centers. TCP is considered as the main transport protocol in data centers. However, TCP has major problems; e.g. TCP reacts upon packet loss event whereas in most data center applications, packet loss causes a huge degradation in performance such as converged SAN, HPC, cloud computing and FCoE. Additionally, TCP consumes, on average, over 20%

of CPU power and at small packet sizes, CPU becomes the bottleneck and cannot saturate the link because of TCP overhead (Zhu *et al.*, 2015).

Other significant challenges that face current congestion control mechanisms are Fairness between flows of different packet sizes and different RTTs. Most network devices detect congestion when their queue level reaches a maximum length in bits, while congestion control mechanisms react per packet. Thus, in traditional congestion control mechanisms, small-packet flows experience a higher number of packet loss than large-packet flows. Therefore, small-packet flows get react by reducing their rates more which result of over-controlling their rates and unfairness issue (Shah *et al.*, 2012; Wilson, 2008). Further, different RTTs strongly reduces the performance of congestion control mechanisms. An experiment is conducted in (Holman *et al.*, 2012) using FreeBSD TCP-NewReno demonstrating that flows with high latency suffer the most when sharing a bottleneck link with low latency flows. Small RTT flows complete more round trips in the same period comparing to large RTT flows which lead to faster recovery. Therefore, short RTT flows get a higher share of the available bandwidth.

To this end, it is widely accepted that Ethernet is the best option for the data center fabric. Therefore, several approaches strive to find a congestion control for Ethernet network. IEEE has recently standardized enhancements to Ethernet in the form of DCB (802.1, 2013) to create a consolidation of I/O connectivity across the data center. The set of standards, defined by the DCB task group within IEEE 802.1 is popularly known as Converged Enhanced Ethernet (CEE) and it comprises PFC (IEEE Standard Association, 2011) and QCN (IEEE 802.1Qau, 2010; Alizadeh *et al.*, 2008) that address congestion control in Ethernet layer. PFC is a link level (hop-by-hop) mechanism that generates and sends PAUSE messages to the sender when the receiver buffer reaches a certain threshold. In contrast, QCN is an end-to-end control mechanism that aims to keep queue length at a predefined level. QCN calculates a feedback value that reflects the severity of congestion and sends this calculated value to the flow responsible for congestion. Yet, PFC and QCN face some issues that we discuss in detail in section 3.3.

Moreover, RoCE v1 (Association *et al.*, 2010) and v2 (Association *et al.*, 2014) are presented as new network protocols which allow performing RDMA over Ethernet network. The reason behind adopting such design is the limitations that face the original design of RDMA over IB, such as the requirement of adopting new network infrastructure which has experienced limited success in enterprise data centers. RDMA technology offers high throughput, low latency, and low CPU overhead, by allowing network interface cards (NICs) to transfer data in-and-out of the host's memory directly. RoCE presents an intermediate layer with IB as an upper interface to support RDMA and Ethernet as a lower interface. This allows using RDMA over standard Ethernet infrastructure with specific NICs that support RoCE. Yet, such protocol requires a congestion control mechanism in Ethernet network (Association *et al.*, 2014).

In this research, we aim to design an Ethernet congestion control mechanism that achieves a robust and reliable data center fabric by achieving i) high link utilization, ii) close-to-zero queue length, iii) low latency, iv) fairness between flows of different packet sizes and v) fairness between flows of different RTTs. We propose HetFlow, an Ethernet end-to-end distributed congestion control mechanism that prevents congestion by controlling hosts' transmission rate. HetFlow is designed with consideration of Ethernet network characteristics that differ from Internet network namely (i) propagation delay is negligible compared to processing delay (few hundred meters), (ii) no per-packet acks, (iii) packet pausing may cause congestion spreading (as in PFC), (iv) Ethernet switches have shallow buffers.

In summary, our main contributions are i) introducing HetFlow as an Ethernet congestion control, ii) proposing a mathematical model for HetFlow, iii) studying the stability and the fairness of HetFlow both mathematically and heuristically. In addition, a testbed implementation of our proposed mechanism is carried out using Intel's Data Plane Development Kit (DPDK) (Intel, 2014). Furthermore, several simulation experiments are conducted for the sake of the comparison between our proposal, TIMELY (Mittal *et al.*, 2015) and QCN (IEEE 802.1Qau, 2010). We chose TIMELY for comparison because both HetFlow and TIMELY are delay-based congestion control protocols. In addition, a comparison with QCN is presented because both HetFlow and QCN are Ethernet end-to-end congestion control protocols.

The rest of this paper is organized as follows. A background on congestion control protocol is presented in section 3.3. Section 3.4 presents our proposed congestion control mechanism. Section 3.5 and 3.6 depict a stability analysis and a scalability study of HetFlow respectively. Section 3.7 depicts the performance evaluation of HetFlow with flows of different RTTs and different packet sizes. A DPDK implementation of HetFlow is presented in section 3.8. Summary and some implementation remarks about our proposal are presented in section 3.9. Related work is presented in section 3.10. Finally, section 3.11 introduces conclusion and future work.

3.3 Background

In this section, we present some research work that is closely related to congestion control in data centers at both Ethernet layer and transport layer. IEEE has recently presented DCB (802.1, 2013) that comprise several enhancements to Ethernet network in data centers. DCB aims to eliminate packet loss due to queue overflow. Ethernet PAUSE IEEE 802.3x and PFC (IEEE Standard Association, 2011) are presented in DCB as link level (hop-by-hop) mechanisms. Ethernet PAUSE was issued to solve packet loss problem by sending a PAUSE request to the sender when the receiver buffer reaches a certain threshold. Thus, the sender stops sending any new frames until a local timer expires or a notification message is received to resume transmission. Ethernet PAUSE is a coarse-grained protocol that causes HOL blocking. PFC is proposed to address Ethernet PAUSE limitations by discriminating data traffic into eight classes that are defined by IEEE 802.1p standard (Ek, 1999). Each class could be controlled individually which reduces HOL blocking, however, PFC does not eliminate HOL blocking (Stephens *et al.*, 2014). In addition, PFC propagates congestion along data path in a phenomenon called tree saturation (Hanawa *et al.*, 1996).

An example explaining the HOL blocking and congestion spreading in PFC is illustrated in Figure 3.1. In this scenario, hosts H11 and H12 are sending data to host H3x and host H1x to H2x. Switch 1 executes ECMP and distributes the traffic on both Spine 1 and Spine 2. The traffic destined to H3x causes congestion at Switch 3 at the output port that is connected to

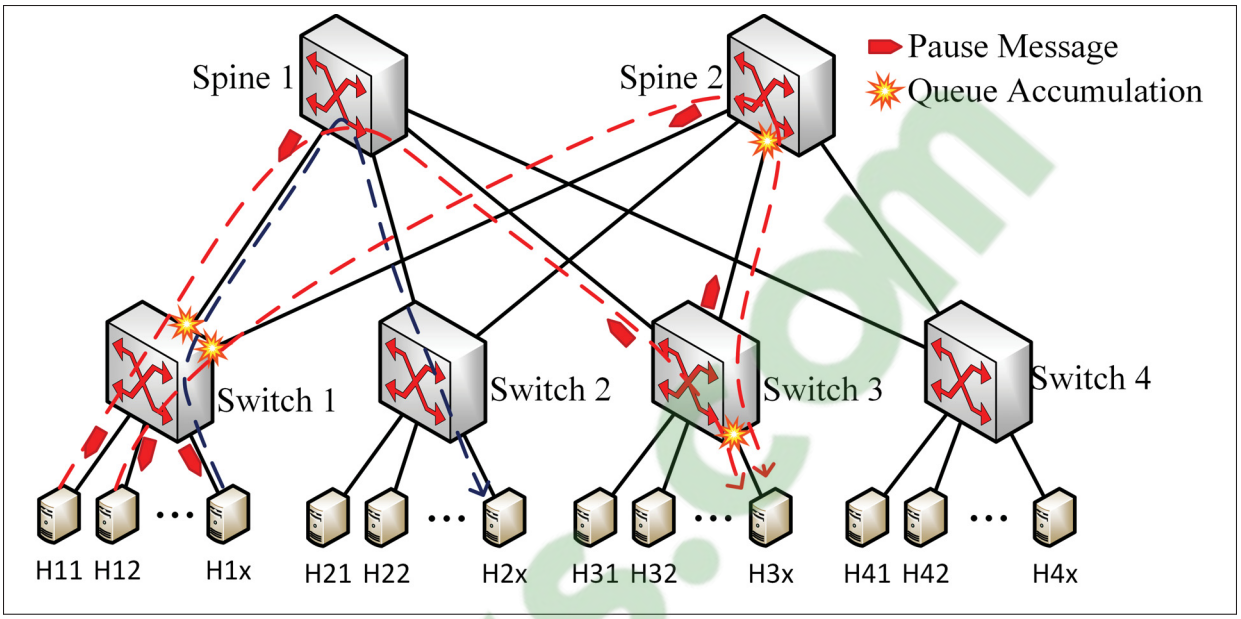


Figure 3.1 PFC HOL Blocking

H3x. Switch 3 reacts by sending pause messages for all adjacent switches/ hosts that transmit data to this port (Spine 1 and Spine 2). The process takes place at Spine 1 and 2 where two pause messages are sent to Switch 1 on both its upward connections. As a final step, Switch 1 reacts by sending pause messages to all adjacent nodes that send traffic to Spine 1 and 2. It is clearly shown that PFC spreads the congestion over the datapath causing what is known as tree saturation (Hanawa *et al.*, 1996) or congestion spreading. In addition, traffic that is originated from host H1x and destined to H2x is throttled at Switch 1 due to a congestion that is originally not in its path. This phenomenon is called HOL blocking.

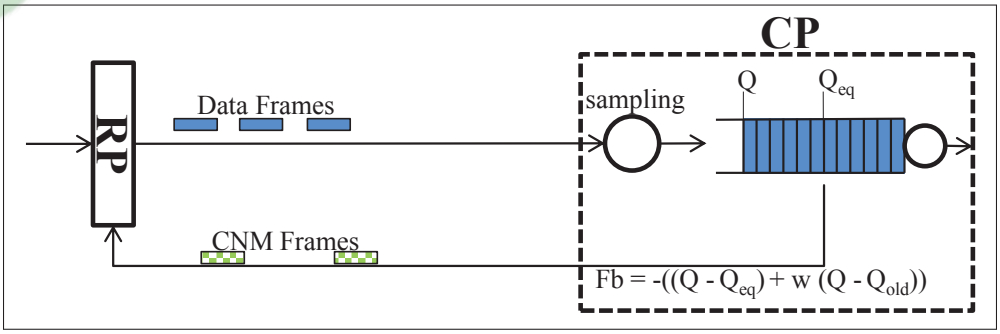


Figure 3.2 QCN framework: CP in the bridge, and RP in the host's NIC

QCN (IEEE 802.1Qau, 2010; Alizadeh *et al.*, 2008) is an end-to-end control mechanism which aims to keep queue length at a predefined level called equilibrium queue length (Q_{eq}). QCN consists of two parts, (i) a CP (in bridges) and (ii) a RP (in hosts) (Fig. 3.2). The CP measures the queue length (Q), and calculates a feedback (Fb) value, in a probabilistic manner, to reflect the congestion severity (Equation 3.1).

$$Fb = -((Q - Q_{eq}) + w \times (Q - Q_{old})). \tag{3.1}$$

Where Q_{old} is the previous queue length and w is a constant which equals 2 (for more details refer to (IEEE 802.1Qau, 2010)). If the calculated Fb is negative, the CP creates a CNM and sends it to the CP. At end host level, when CP receives CNM, it decreases its transmission rate accordingly. If no CNM is received, the CP increases its transmission rate according to a three-phase rate increase algorithm (IEEE 802.1Qau, 2010).

Due to the probabilistic manner of calculating Fb, QCN experiences several issues regarding fairness (Kabbani *et al.*, 2010; Zhang & Ansari, 2013) and queue length fluctuation (Tanisawa & Yamamoto, 2013). Moreover, both PFC and QCN functionalities are deeply integrated into switch ASICs that requires switch modification which we aim to avoid.

TIMELY (Mittal *et al.*, 2015) is a delay-based congestion control scheme for data centers. It uses the deviation of RTT to identify the congestion. TIMELY relies on the capability of NIC hardware to obtain fine-grained RTT measurements. In TIMELY, the receiver sends an acknowledge per segment of data of size 16 - 64 KB. At the sender, Upon receiving an ACK, RTT is calculated and the gradient of RTT in order to control the transmission rate. TIMELY defines RTT as the propagation and queuing delay only. Thus, segment serialization time (time to put the segment on the wire) is subtracted from the completion time in order to calculate RTT as shown in Fig. 3.3.

TIMELY is a rate-based protocol that computes a new rate after receiving each ACK based on RTT value as follow:

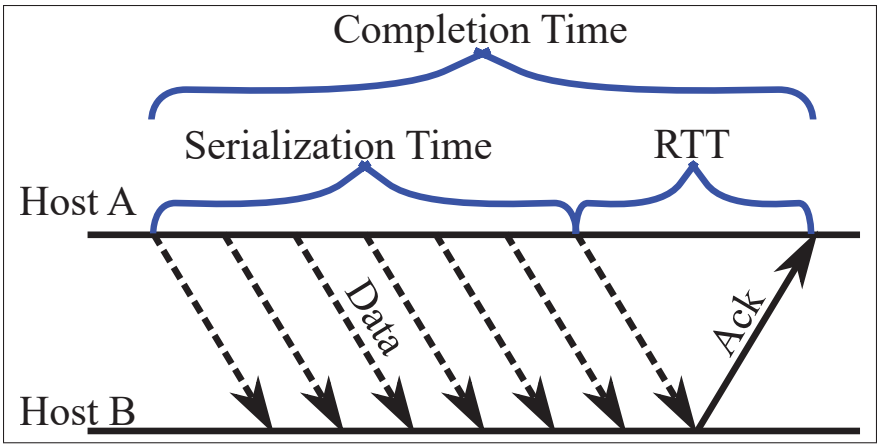


Figure 3.3 RTT calculation in TIMELY

i. If $RTT < T_{low}$, TIMELY increases transmission rate R additively by a constant δ.

ii. If $RTT > T_{high}$, TIMELY decreases R multiplicatively by a factor β.

iii. if $T_{low} \leq RTT \leq T_{high}$, TIMELY calculates the gradient of RTT, $g = \frac{RTT - RTT_{old}}{D_{minRTT}}$ and controls the transmission rate using (3.2).

$$\begin{cases} R \leftarrow R + \delta & \text{If } RTT < T_{low} \\ R \leftarrow R \times (1 - \beta\ (1 - \frac{T_{high}}{RTT}) & \text{If } RTT > T_{high} \\ \begin{cases} R \leftarrow R + N \times \delta & \text{If } g \leq 0 \\ R \leftarrow R \times (1 - \beta \times g) & \text{If } g > 0 \end{cases} & \text{Otherwise.} \end{cases} \tag{3.2}$$

TIMELY uses a per-packet pacing to apply the newly calculated rate.

3.4 HetFlow: Heterogeneous Flow congestion control mechanism

In this section, we give a detailed description of the HetFlow architecture and we explain the interactions between its internal components. HetFlow is a delay-based congestion control mechanism for Ethernet networks. It is a distributed algorithm that runs on end-hosts without the need for switch participation. HetFlow recognizes congestion by detecting the variation of one-way delay (OWD), then it sends back a CNM to the source responsible for the congestion

to reduce its transmission rate. Table 3.1 lists all HetFlow notations that is used in the rest of the paper.

BC_r	Sampling byte counter
C	Link Capapcity
dv	Delay variation
Fb	Feedback value
$\widehat{Fb}$	The last executed feedback value
G_d	A constant
KC	Scaling factor
L	Frame size
N	Number of flows
N_s	Sample size in frames
OWD	One-way delay per frame
owd	Average OWD per sample
owd_{old}	Average OWD of the previous sample
$prop\_delay$	The propagation delay
Q_{req}	The required buffer capacity
R	transmission rate
$\widehat{R}$	The target rate for the recovery phase
t	time
$\widehat{t}$	The execution time of the last CNM
T	Sampling timer
t_{tr}	Frame transmission time
t_{tr_next}	Next frame transmission time
η	The period between two samples
σ	The recovery time
τ	The propagation delay

Table 3.1 HetFlow notations

3.4.1 HetFlow Components

HetFlow comprises four components namely (i) *Time Stamper*, (ii) *Rate Pacing*, (iii) *Data Sampler*, and (iv) *CNM Generator* as depicted in Fig. 3.4. HetFlow operates on a per-flow basis while a flow is defined as source-MAC/Destination-MAC/priority triple. The functionality of these components is detailed as follows.

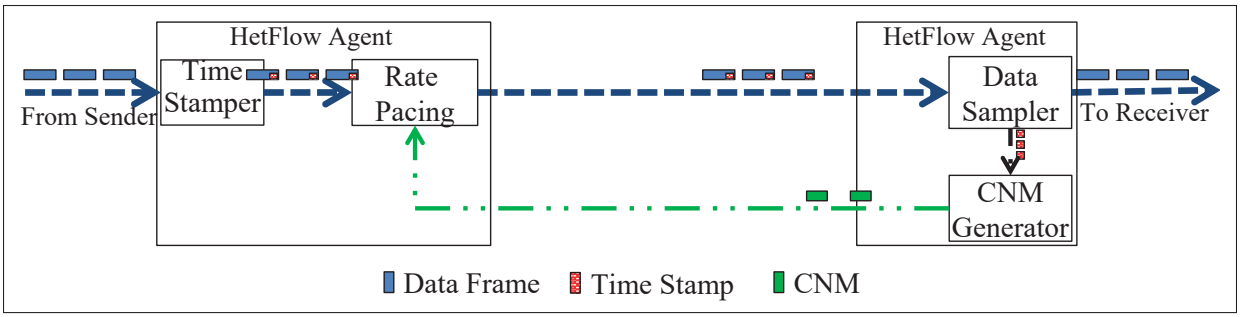


Figure 3.4 HetFlow components

HetFlow control cycle starts by adding a timestamp to data packets using the *Time Stamper* module[1].

At the receiver side, the *Data Sampler* samples the received data based on a byte counter (BC_r) and a timer T whichever expires first. Sampling based on byte counter forces HetFlow to have a bit-based reaction, therefore, it achieves fairness between flows of different packet sizes. In addition, the timer T is used to activate the congestion reaction for large number of low-rate flows and the byte counter did trigger the reaction process. The *Data Sampler* extracts time stamp information of N_s packets. Next, the time stamp information of those N_s packets is acquired and then relayed to the *CNM Generator*. Thereafter, HetFlow *CNM Generator* calculates the following parameters: (i) the delay for each packet within the sample, (ii) *owd*, the average OWD within each sample and (iii) *dv*, the delay variation related to the previous sampling period using (3.3).

$$\begin{cases} dv = \frac{owd - owd_{old}}{\eta} \\ \eta = min(T, R/BC_r). \end{cases} \tag{3.3}$$

Where owd_{old} is the weighted moving average (EWMA) of OWD of the previous sample, and η is the period between two samples which is a function BC_r. Therefore, dividing the delay variation dv by η cancels its dependency on the packet size which produces a congestion metric

[1] the timestamp can be sent within the packet as meta-data, for example as a header extension at the IPv6 level, in the option field at IPv4 level, or in an extra field in the Ethernet frame header as a new standard.

that reflects flow rate regardless of its packet sizes. By doing so, HetFlow congestion control mechanism endeavors to achieve fairness between flows of different packet sizes. Ultimately, the HetFlow *CNM Generator* calculates a congestion metric, so-called feedback Fb, using (3.4). If Fb is greater than zero, a CNM message is sent back to the *Rate Pacing*. If a CNM message is lost on its way back to the source, the HetFlow *CNM Generator* detects that the host is not reacting, and it generates another CNM message. This comes at the cost of queue length increasing but thanks to the early reaction of HetFlow, queue length increase will not be noticeable.

$$Fb = KC \times dv. \tag{3.4}$$

Where KC is a scaling factor that is used to scale the dv values to the Fb range as in QCN ($[-64, 64]$). Congestion detection pseudo code is shown in algorithm 3. As the algorithm depicts, parameter initialization is carried out at line 2. For each received packet, reading time stamp information and updating OWD and BC_r are carried out at lines 4 - 6. At line 7, HetFlow verifies if either the timer or the byte counter expired, then it starts sampling data frames and calculates owd of N_s packets (lines 7). Once these N_s packets are received, the average OWD, dv, and Fb are calculated in lines 9 to 14. Further, for positive Fb, HetFlow sends a CNM back to the source at line 16. After each sample, HetFlow reinitializes its parameters at lines 18 - 19.

HetFlow uses a rate-based control scheme instead of a window-based one. The latter faces several hurdles particularly with the rapid increase of propagation delay to the transmission time ratio in today's networks (Charny *et al.*, 1995; Jain, 1998). HetFlow *Rate Pacing* reacts upon receiving CNM by extracting Fb value and reducing the transmission rate R based on (3.5).

$$\begin{cases} \widehat{R} \leftarrow R \\ R \leftarrow R(1 - Gd \times Fb). \end{cases} \tag{3.5}$$

Algorithm 3: Congestion detection and Fb calculation process

```
1  initialization;
2  BC_r = 0; sample_time = current_time; Timer ← T ;
3  foreach Received frame do
4      OWD ← receiving_time − sending_time ;
5      len ← read(packetlength) ;
6      BC_r += len;
7      if ((BC_r > BC_Limit) OR Timer expired) then
8          d_sum += OWD;
9          if N_s packets received then
10             owd = d_sum/N_s;
11             η = current_time − sample_time;
12             dv = (owd − owd_old)/η;
13             owd_old = EWMA(owd);
14             Fb = KC × dv;
15             if (Fb > 0) then
16                 send CNM;
17             end
               /* Parameter reinitialization                    */
18             BC_r = 0;
19             sample_time = current_time;
20     end
21 end
```

Where Gd is a constant taken so that $Gd \times Fb_{max} = 1/2$, i.e. the transmission rate can decrease by 50% in the worst case, and $\widehat{R}$ is the target rate for the recovery phase which corresponds to the transmission rate before congestion. Moreover, when the HetFlow *Rate Pacing* does not receive any CNM message within a period defined by a timer (T), it executes a rate increase process inspired by CUBIC TCP (Ha *et al.*, 2008) (Equation 3.6).

$$R \leftarrow \widehat{R}\left(1 + Gd \times \widehat{Fb} \times \left(\frac{t - \widehat{t} - \sigma}{\sigma}\right)^3\right). \tag{3.6}$$

Where t is the current time, $\widehat{Fb}$ is the last executed feedback value, $\widehat{t}$ is the execution time of the last CNM, and σ is the recovery time which defines how fast the flows recover from congestion (Fig. 3.5).

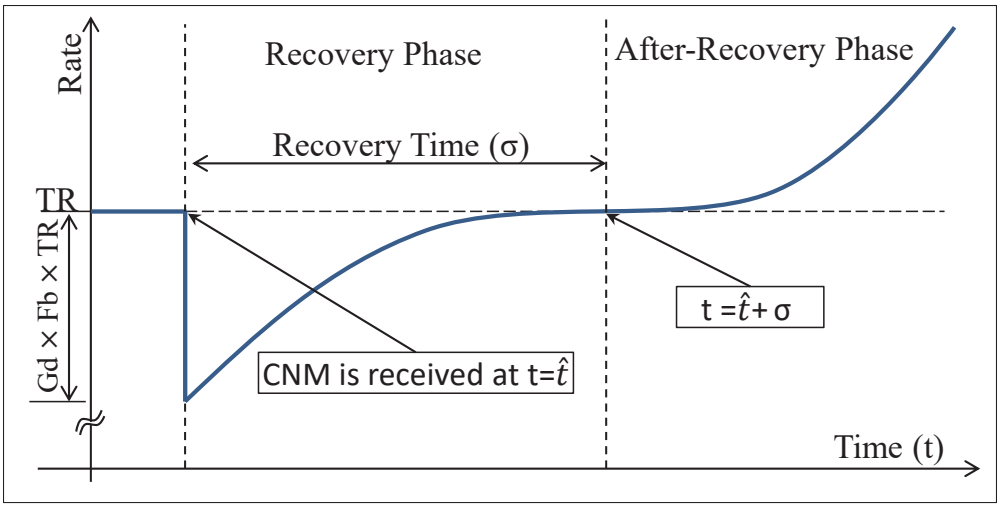


Figure 3.5 HetFlow rate control operation

Based on (3.6), one can notice that HetFlow divides rate increase process into two phases namely (i) recovery phase, and (ii) after-recovery phase (Fig. 3.5). In the beginning of the recovery phase, HetFlow increases the transmission rate R rapidly. Then, HetFlow slows down the increase process as it approaches $\widehat{R}$. Finally, when HetFlow passes $\widehat{R}$ successfully and reaches the after-recovery phase, it starts increasing R rapidly again. One can notice that the formula does not depend on RTT, which eliminates the effect of RTT on HetFlow performance; therefore, it achieves fairness between flows of different RTTs.

Algorithm 4 presents the pseudo code of HetFlow's rate decreasing processes. Once HetFlow receives a positive Fb, it updates $\widehat{R}$, and decreases the rate R by a factor that is equal to $(1 - G_d \times Fb)$ at lines 4 and 5 respectively. At lines 6 and 7, $\widehat{t}$ and $RateIncreaseActive$ are set. They will be used later to start executing the rate increase process (Algorithm 5).

Algorithm 5 depicts the pseudo code of HetFlow's rate increase processes. It verifies if the rate increase process is active at line 2 ($RateIncreaseActive == TRUE$), then it executes the rate update (line 3). In addition, it verifies, if the maximum local link capacity is reached, then it disables the rate increase process (lines 4 to 6).

Algorithm 4: HetFlow rate decrease process

```
1 foreach CNM message do
2     read(Fb) ;
3     if (Fb ≥ 0) then
4         R̂ ← R ;
5         R ← R(1 − G_d × Fb) ;
          /* Start rate increase process                    */
6         t̂ ← current_time ;
7         RateIncreaseActive ← TRUE ;
8     end
9 end
```

Algorithm 5: HetFlow rate increase process

```
1 foreach timeout do
2     if RateIncreaseActive == TRUE then
3         R ← R̂(1 + Gd × Fb × ((t − t̂ − σ)/σ)^3) ;
4         if (R > linkCapacity) then
5             R ← linkCapacity ;
6             RateIncreaseActive ← FALSE ;
7         end
8     end
9 end
```

3.4.2 HetFlow Model

In this section, we develop a fluid model for HetFlow and validate the model through simulation. In this model, HetFlow calculates the feedback value using (3.7).

$$\begin{aligned} Fb(t) &= Kc\frac{owd(t) - owd(t-\eta)}{\eta} \\ &= Kc \times owd'(t-\eta) \end{aligned} \tag{3.7}$$

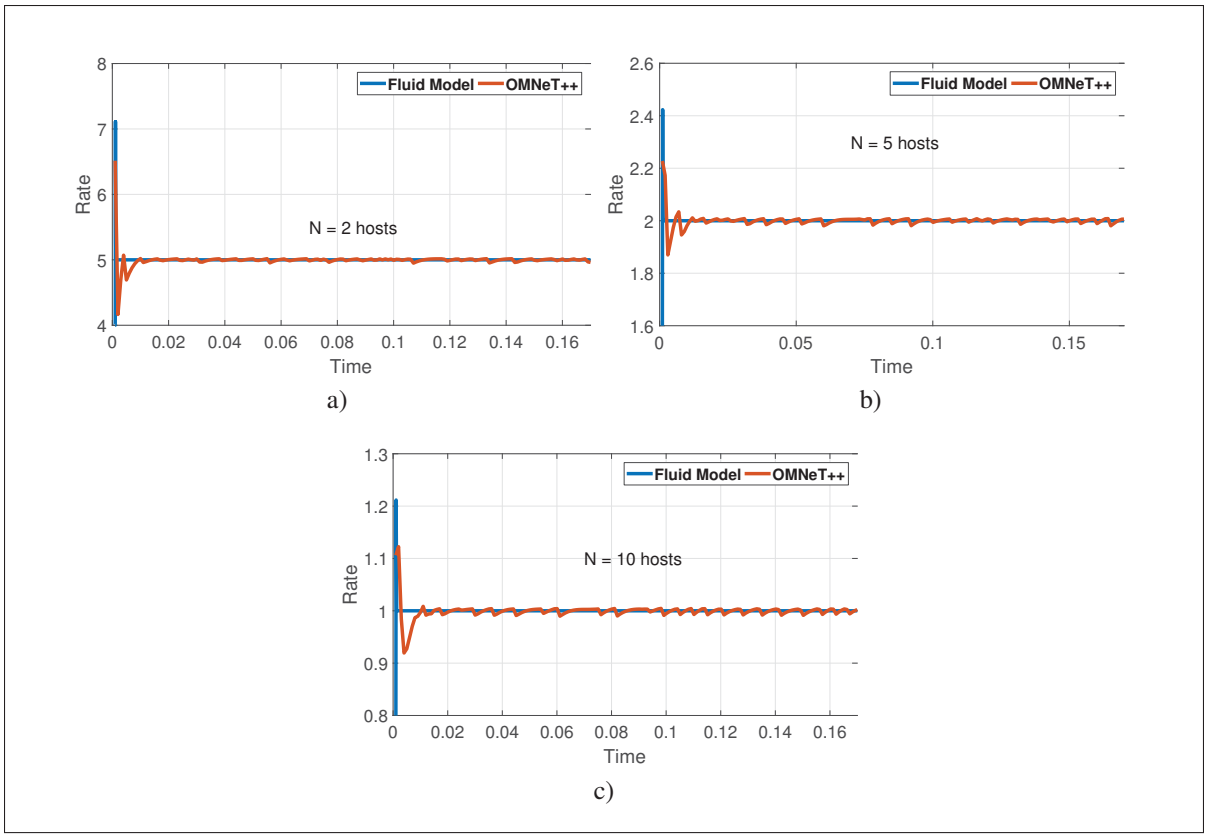


Figure 3.6 Comparison of HetFlow fluid model and OMNeT++ simulations

Where $owd(t)$ is represented as a function of propagation delay $prop\_delay$ by (3.8).

$$\begin{aligned} owd(t) &= \frac{1}{C} \times q(t) + L/C + prop\_delay \\ &= \frac{1}{C} \int (N \times R(t) - C)\, dt + L/C + prop\_delay \end{aligned} \tag{3.8}$$

Thus,

$$owd'(t) = \frac{N}{C} R(t) - 1 \tag{3.9}$$

Therefore, feedback could be represented by (3.10):

$$Fb(t) = Kc \times \left(\frac{N \times R(t-\eta)}{C} - 1 \right) \tag{3.10}$$

Hence, HetFlow rate decrease subsystem equation (3.5) and rate increase subsystem equation (3.6) can be represented by (3.11) and (3.12) respectively.

$$\begin{cases} \widehat{R} \leftarrow R(t) \\ \widehat{t} = t \\ \widehat{Fb} = Kc \times \left(\frac{N \times R(t-\eta-\tau)}{C} - 1\right) \\ R(t) \leftarrow R(t) \times \left(1 - Gd \times Kc\left(\frac{N}{C}R(t-\eta-\tau) - 1\right)\right) \end{cases} \quad \text{If } Fb \geq 0 \tag{3.11}$$

$$R(t) \leftarrow \widehat{R} \times \left(1 + Gd \times \widehat{Fb} \times \left(\frac{t - \widehat{t} - \sigma}{\sigma}\right)^3\right) \quad \text{If } Fb < 0 \tag{3.12}$$

where τ is the propagation delay. Therefore, we induce the delay differential equations for HetFlow system as follow:

$$\begin{cases} \frac{dR(t)}{dt} = \frac{-Gd \times Kc}{\eta}\left(\frac{N}{C}R(t-\eta-\tau) - 1\right) \times R(t) & \text{If } Fb \geq 0 \\ \frac{dR(t)}{dt} = \frac{3 \times Gd \times \widehat{Fb}}{\sigma^3} \times \left(t - \widehat{t} - \sigma\right)^2 \times \widehat{R} & \text{Otherwise} \end{cases} \tag{3.13}$$

Fig. 3.6 shows the comparison results of the HetFlow fluid model and the HetFlow simulation, for a different number of flows (N = 2 , 5, and 10 hosts). It shows that our proposed fluid model matches the simulation one.

3.5 HetFlow Stability

In this section, we study the stability of our proposal.

3.5.1 HetFlow Stability Analysis

Lemma 3.1. *HetFlow has a unique fixed point of flow rates $R = C/N$, and $R' = 0$.*

Proof. We first obtain the fixed point of the system and study the derivative of the rate around this fixed point. For any fixed point of HetFlow (if they exist) the derivative of the rate R' and one-way delay owd' must be zero. By setting the left hand side of (3.9) to zero, we can notice that it satisfies the fixed point criteria where $R = C/N$ and $Fb = 0$. In addition, by substituting $R = C/N$ in 3.13, we get $R' = 0$. Therefore, at the fixed point, the one-way delay become stable ($owd' = 0$) and flow rates become stable at $R = C/N$.

□

Lemma 3.2. *For N flows, HetFlow control function converges to the stable point* $(R = R^* = C/N, R' = 0)$.

Proof. In order to prove the stability of HetFlow's differential equations, we assert that for any function $f(R) = \frac{d\ R(t)}{dt}$, the derivative around its fixed point approaches to zero; i.e. it is positive for all $R < R^*$ and negative for all $R > R^*$, where R^* is the fixed point value. Therefore, for all values of y, HetFlow converges to the stable point R^*. Equation 3.14 represents this concept mathematically.

$$\begin{cases} f(R) - f(R^*) & < 0 \quad \text{If } R > R^* \\ f(R) - f(R^*) & > 0 \quad \text{If } R < R^* \end{cases} \tag{3.14}$$

By dividing equation (3.14) by $(\Delta R = (R - R^*))$, we get:

$$\frac{f(R) - f(R^*)}{R - R^*} < 0 \quad \text{For all } R \tag{3.15}$$

Considering that the derivative of a function $\frac{d\ f(R)}{dR} \approx \frac{\Delta f(R)}{\Delta R}$, hence, inequality (3.15) becomes:

$$\frac{d\ f(R)}{dR} < 0 \tag{3.16}$$

Therefore, by asserting that inequality (3.16) is satisfied around the fixed point, we prove that the system is stable.

For the rate decrease subsystem, we get the derivative $\frac{dR'}{dR}$ of rate decrease part of (3.13) and calculate its value at the fixed point as follow:

$$\frac{dR'}{dR} = \frac{d\Big(\frac{-Gd\times Kc}{\eta} \times \frac{N}{C}\big(R(t-\eta-\tau)-1\big) \times R(t)\Big)}{dR} \tag{3.17}$$

We use Taylor series to approximate (3.17) and take the first two terms.

$$\begin{aligned}
\frac{dR'}{dR} &\approx \frac{d\Big(\frac{-Gd\times Kc\times N}{\eta\times C}\big(R-(\eta+\tau)R'-1\big)\times R\Big)}{dR} \\
&= \frac{-Gd\times Kc\times N}{\eta\times C}\Big(\big(R-(\eta+\tau)R'-1\big)+ \\
&\quad R(1-(\eta+\tau)\frac{dR'}{dR})\Big) \\
&= \frac{-Gd\times Kc\times N}{\eta\times C}\Big(\big(C/N-(\eta+\tau)0-1\big)+ \\
&\quad C/N\times(1-(\eta+\tau)\frac{dR'}{dR})\Big) \\
&= \frac{-Gd\times Kc}{\eta}\Big(1-(\eta+\tau)\frac{dR'}{dR}\Big) \\
&= -Gd\times Kc/\eta + \big(Gd\times Kc(\eta+\tau)/\eta\big)\frac{dR'}{dR} \\
&= \frac{-Gd\times Kc/\eta}{1-Gd\times Kc(\eta+\tau)/\eta} \\
&= \frac{-Gd\times Kc}{\eta-Gd\times Kc(\eta+\tau)} \\
&= \frac{-Gd\times Kc}{(1-Gd\times Kc)\eta-Gd\times Kc\times\tau} \\
&= \frac{-1}{(\frac{1}{Gd\times Kc}-1)\eta-\tau}
\end{aligned} \tag{3.18}$$

From the design of HetFlow $Gd\times Kc \leq 0.5$, therefore, $1/(Gd\times Kc) \geq 2$. Hence, one can conclude that for flow rates that are greater than the fixed point $(R = C/N, R' = 0)$ and $\eta > \tau$, $\frac{dR'}{dR} < 0$ and HetFlow converges towards the fixed point (the right side of Fig. 3.7).

For the rate increase subsystem in (3.13), we calculate the derivative $\frac{dR'}{dR}$ at the fixed point.

$$\begin{aligned} \frac{dR'}{dR} &= \frac{d\left(\frac{3\times Gd\times \widehat{Fb}}{\sigma^3}\times (t-\widehat{t}-\sigma)^2\times \widehat{R}\right)}{dR} \\ \frac{dR'}{dt}\Big/\frac{dR}{dt} &= \frac{\frac{6\times Gd\times \widehat{Fb}}{\sigma^3}\times \widehat{R}\times (t-\widehat{t}-\sigma)}{\frac{3\times Gd\times \widehat{Fb}}{\sigma^3}\times \widehat{R}\times (t-\widehat{t}-\sigma)^2} \\ &= \frac{2}{(t-\widehat{t}-\sigma)} \end{aligned} \tag{3.19}$$

One can notice that during the recovery phase where $t-\widehat{t}<\sigma$, $\frac{dR'}{dR}<0$ and HetFlow converges to the fixed point (the left side of Fig. 3.7). Otherwise, when $t-\widehat{t}>\sigma$, the system switches to the right side and moves left towards the fixed point. Therefore, HetFlow system is stable around the fixed point $(R=C/N, R'=0)$.

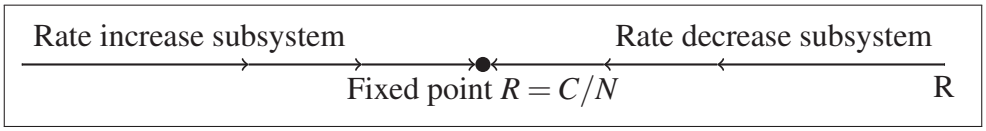


Figure 3.7 HetFlow convergence around its fixed point.

□

3.5.2 HetFlow Stability Evaluation

To evaluate the performance of the proposed mechanism, we use OMNeT (Varga, András and Hornig, Rudolf, 2008) to build a simulation model for HetFlow. In our implementation, HetFlow parameters are set as shown in Table 3.2 unless otherwise mentioned. Besides, data sources send variable packet sizes based on normal distribution (Average = 600 Bytes, standard deviation = 150). Several experiments are conducted in this simulation environment to evaluate the performance of HetFlow and to compare it with TIMELY and QCN.

TIMELY and QCN parameters are set as in (Mittal *et al.*, 2015) and (Alizadeh *et al.*, 2008) respectively [2]. Further, in this simulation, a per-packet pacing is used to apply the newly calculated rate.

Table 3.2 Simulation parameters

HetFlow Parameters	
Rate controller timer	$T = 3ms$
Sample size	$N_s = 32$ packets
Receiver byte counter	$BC_r = 100$ KByte
Scaling factor	$KC = 1500$
Recovery time	$\sigma = 10ms$
Data Senders Parameters	
Frame size	Normal distribution ($avg = 600, \sigma = 150$)
Min Frame size	200 Bytes
Max Frame size	1500 Bytes

The simulation is conducted on a dumbbell topology as shown in Fig.3.8. In this topology, N data sources send data to N receivers. Hosts start sending at 80% of the bottleneck link capacity and increase linearly till the link becomes saturated. All hosts are connected to the switches using 10-Gbps links.

Figure 3.9 shows the simulation results for a different number of flows (N = 4, 10) while using HetFlow, QCN and TIMELY. The figure shows that HetFlow achieves fairness between flows (Fig. 3.9a and 3.9d). One can notice that due to the probabilistic behavior of QCN, fairness between flows is difficult to achieve (Fig. 3.9b and 3.9e). These results are corroborated by

[2] TIMELY parameters: $\beta = 0.8$, $\alpha = 0.875$, $T_{low} = 50\mu s$, $T_{high} = 500\mu s$, $D_{minRTT} = 20\mu s$. QCN parameters: $Q_{eq} = 20\%$ of the maximum queue length and $Gd = 1/128$

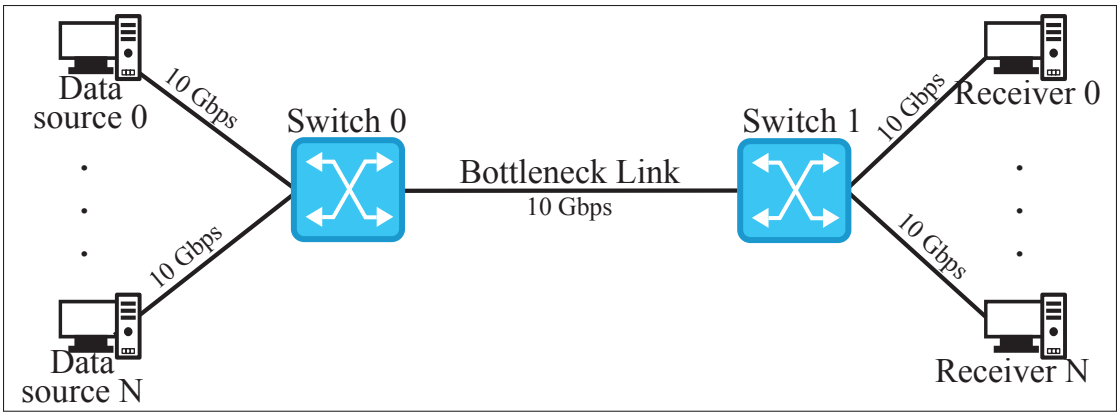


Figure 3.8 Simulation topology

the findings of (Zhang & Ansari, 2013; Kabbani *et al.*, 2010). Fig. 3.9c and 3.9f show that TIMELY could not achieve fairness between flows because it does not have a fixed point. Indeed, it is shown in (Zhu *et al.*, 2016) that TIMELY has multiple fixed point. HetFlow, QCN and TIMELY are able to control the cross traffic as shown in figures 3.9g and 3.9i. However, HetFlow outperforms both QCN and TIMELY in terms of queue length stability as shown in figure 3.9h and 3.9j.

3.6 HetFlow Scalability

In this section, we study the scalability of HetFlow mathematically, and we use simulations to verify our mathematical analysis.

3.6.1 HetFlow Scalability Analysis

HetFlow scalability is controlled mainly by the required buffer capacity as the number of flows grow. Therefore, we study the minimum required buffer capacity Q_{req} for HetFlow in order to prevent packet loss.

Lemma 3.3. *For HetFlow, the minimum required buffer capacity is proportional to* $\propto (N \times \widehat{R} - C)$ *which approaches zero at the fixed point. Therefore, flow number has a minor effect on HetFlow stability.*

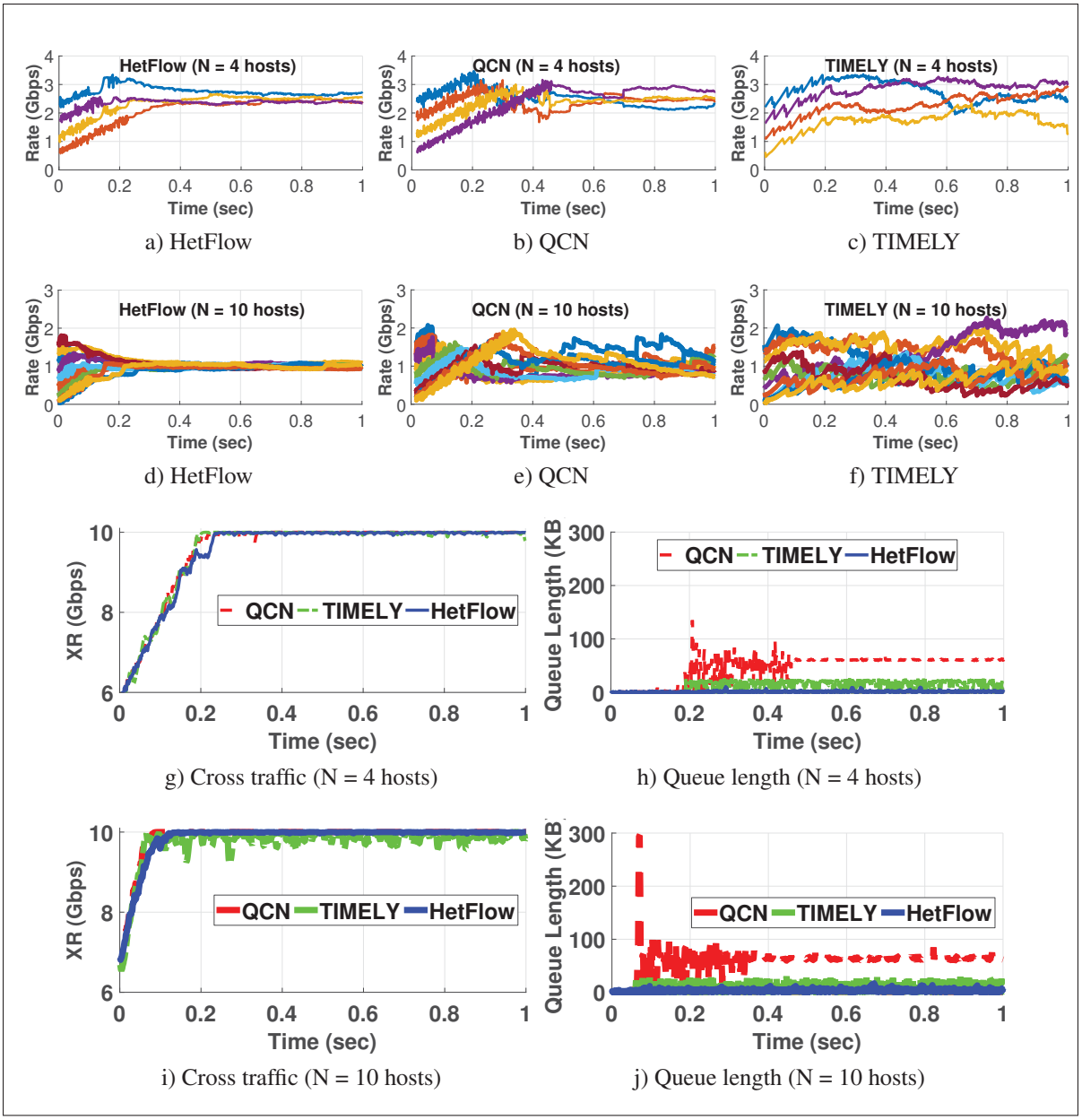


Figure 3.9 Transmission rate for N = 4 and 10 hosts

Proof. Q_{req} must be greater than data accumulation in one control cycle as represented by (3.20).

$$Q_{req} \geq \int_{t=\widehat{t}}^{t=\widehat{t}+\eta+\tau} (N \times R(t) - C)\, dt \tag{3.20}$$

By substituting (3.11) in (3.20), and substituting $T = (\eta + \tau)$, we get:

$$\begin{aligned} Q_{req} &\geq \int_{\widehat{t}}^{\widehat{t}+T} (N \times Gd \times \widehat{Fb} \times \widehat{R} \times \Big(\frac{t - N \times \widehat{t} - \sigma}{\sigma}\Big)^3 + \\ &\quad (N \times \widehat{R} - C))\, dt \\ &\geq \frac{N \times Gd \times \widehat{Fb} \times \widehat{R}}{\sigma^3} (t - \widehat{t} - \sigma)^3 \Big|_{\widehat{t}}^{\widehat{t}+T} + (N \times \widehat{R} - C) \\ &\geq \frac{N \times Gd \times \widehat{Fb} \times \widehat{R}}{4\sigma^3} \Big((T - \sigma)^4 - (-\sigma)^4\Big) + \\ &\quad (N \times \widehat{R} - C) \times (T) \\ &\geq \frac{N \times Gd \times \widehat{Fb} \times \widehat{R}}{4\sigma^3} \Big(T^4 - 3T^3\sigma + 6T^2\sigma^2 - 4T\sigma^3\Big) + \\ &\quad (N \times \widehat{R} - C) \times (T) \end{aligned} \tag{3.21}$$

At the beginning of the rate increase process, $N \times \widehat{R} \geq C$. Thus, after substituting that in (3.21), we get:

$$\begin{aligned} Q_{req} \geq \frac{Gd \times \widehat{Fb} \times C}{8\sigma^3} \Big(T^4 - 3T^3\sigma + 6T^2\sigma^2 - 4T\sigma^3\Big) + \\ (N \times \widehat{R} - C) \times (T) \end{aligned} \tag{3.22}$$

One can notice that the first part of the right hand side of (3.22) does not depend on the number of flows. Therefore, the minimum required buffer capacity Q_{req} depends mainly on the second part of the right hand side of the equation namely $(N \times \widehat{R} - C)$ which approaches zero as the system approaches the fixed point where $\widehat{R} \approx C/N$. Therefore, the number of flows has a minor effect on the HetFlow's required buffer capacity.

□

3.6.2 HetFlow Scalability Evaluation

In this experiment, we evaluate the scalability of HetFlow by increasing the number of flows to 38 in the simulation while using the same dumbbell topology depicted in Fig. 3.8. The

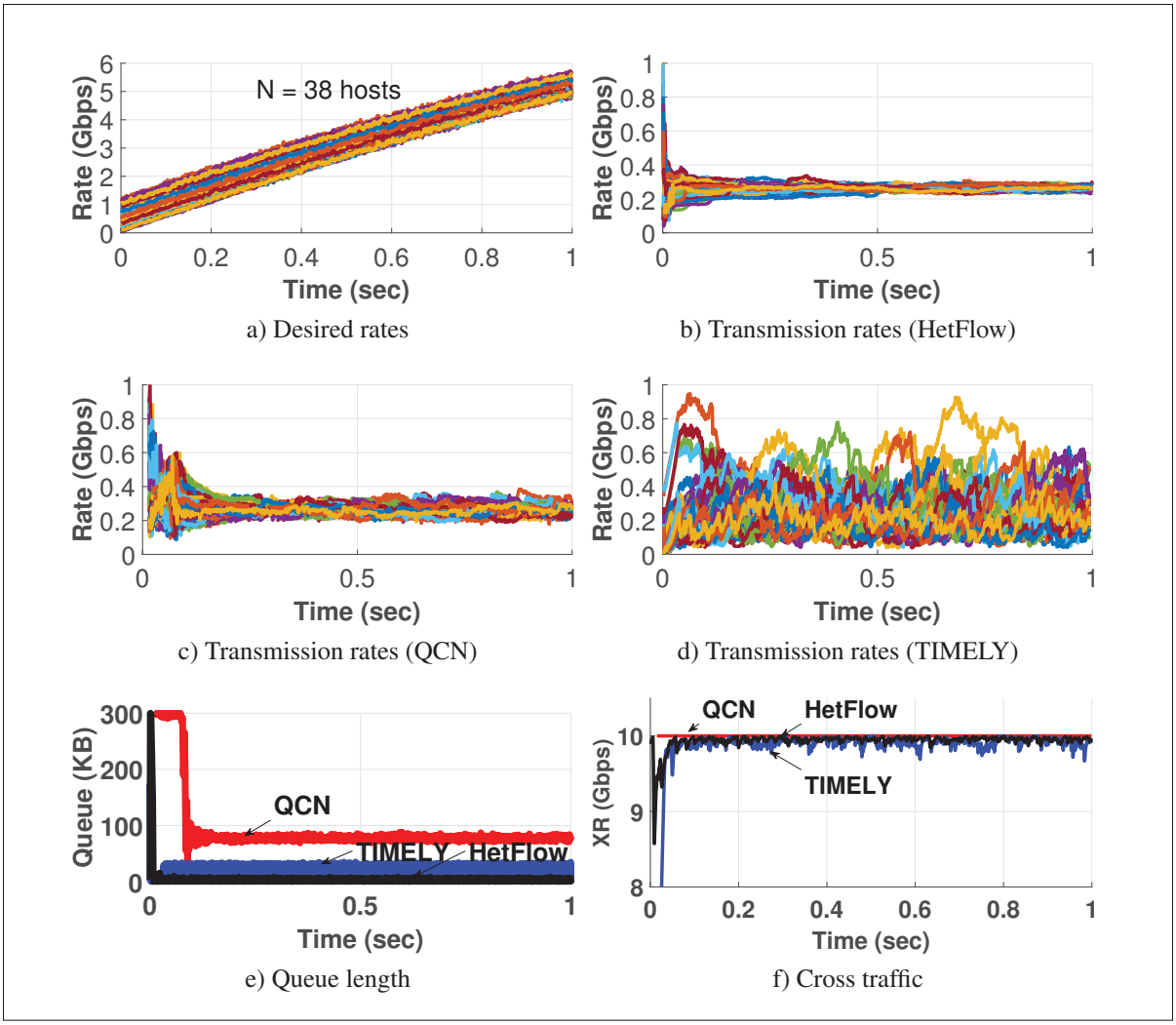
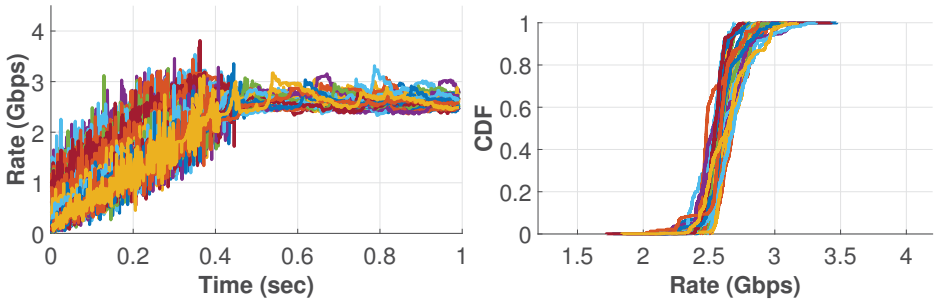


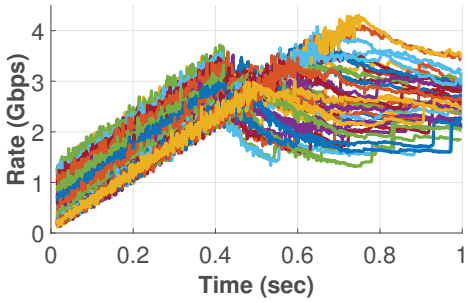
Figure 3.10 HetFlow scalability evaluation (38 hosts in a 10-Gbps network)

number of flows in this experiment was chosen to keep HetFlow sampling process controlled by the byte counter BC_r (bit-based controlled) which achieves batter fairness. Increasing the number of flow further triggers the sampling process by the timer T which reduces the fairness while keeping the congestion under control. In this experiment, we ran the simulation twice, one for a 10-Gbps network and another for a 100-Gbps network. The senders send data with the desired rates shown in Fig. 3.10a. In order to endure a high number of low-rate flows, we set the receiver byte counter $KC = 20$ KB. Fig. 3.10b shows that HetFlow scaled successfully to this number of flows and controlled the transmission rates to prevent congestion. Fig. 3.10c

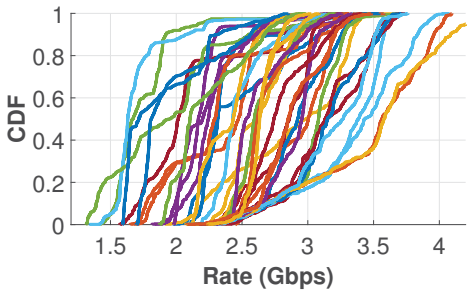


a) Transmission rates (HetFlow)

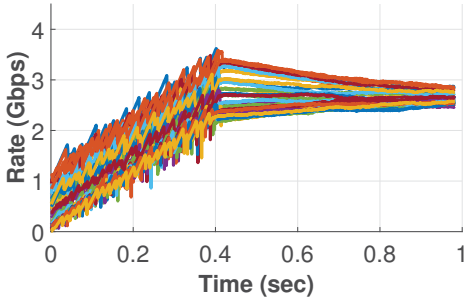
b) CDF (HetFlow)



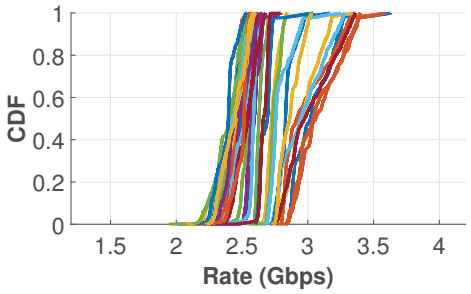
c) Transmission rates (QCN)



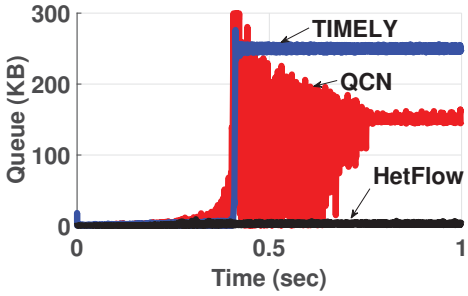
d) CDF (QCN)



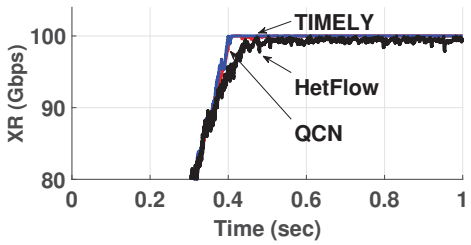
e) Transmission rates (TIMELY)



f) CDF (TIMELY)



g) Queue length



h) Cross traffic

Figure 3.11 HetFlow scalability evaluation (38 hosts in a 100-Gbps network)

and Fig.3.10d depict the transmission rates while using QCN and TIMELY respectively. One can notice that HetFlow outperforms both QCN and TIMELY in terms of fairness between flows. In addition, Fig. 3.10f shows that HetFlow, QCN and TIMELY achieve cross traffic XR close to the maximum link capacity. However, Fig. 3.10e depicts that QCN causes queue saturation for a long period before stabilizing the queue length. Whereas, HetFlow succeeded in maintaining the queue at a close-to-zero level after a narrow peak at the starting time which matches our analytic prediction. TIMELY, on the other hand, causes queue fluctuation around the equivalent of T_{low} which raises the following issue. TIMELY performance depends on its parameter tuning which means that TIMELY requires fine-tuning based on network topology. Hence, we conclude that HetFlow outperforms QCN and TIMELY in terms of queue stability and fairness between flows.

Further, the same experiment is repeated using a 100-Gbps network and the same desired rate (Fig. 3.10a). In order to stabilize QCN queue in this environment, we had to increase Q_{eq} to 50% of the maximum buffer size which increases the probability of picking the culprit flow within the sample. In addition, we set $T_{high} = 250\mu s$ for TIMELY in order to limit the queue overflow and prevent packet loss. The obtained results are illustrated in Fig. 3.11. It is clearly shown that HetFlow achieves better fairness between flows (Fig. 3.11a, Fig.3.11b. Fig. 3.11e reveals that TIMELY achieves a relatively good fairness. However, after plotting the cumulative distribution function (CDF) of the transmission rates during the stable period ($time = 0.4sec - 1sec$), it becomes clear that HetFlow achieves better fairness (Fig. 3.11b). On the other hand, Fig. 3.11f depicts that TIMELY achieves a better fairness than QCN (3.11d). However, TIMELY induces high queue length as shown in Fig. 3.11g. In contrast, HetFlow had fulfilled our analytical prediction and succeeded in achieving a close-to-zero queue length which induces faster response time, and minimum network latency.

3.7 Performance Evaluation

One of our main claims is the ability of HetFlow to achieve fairness between flows of different RTTs and different packet sizes. Dissimilar to TCP-like protocols that face this unfairness

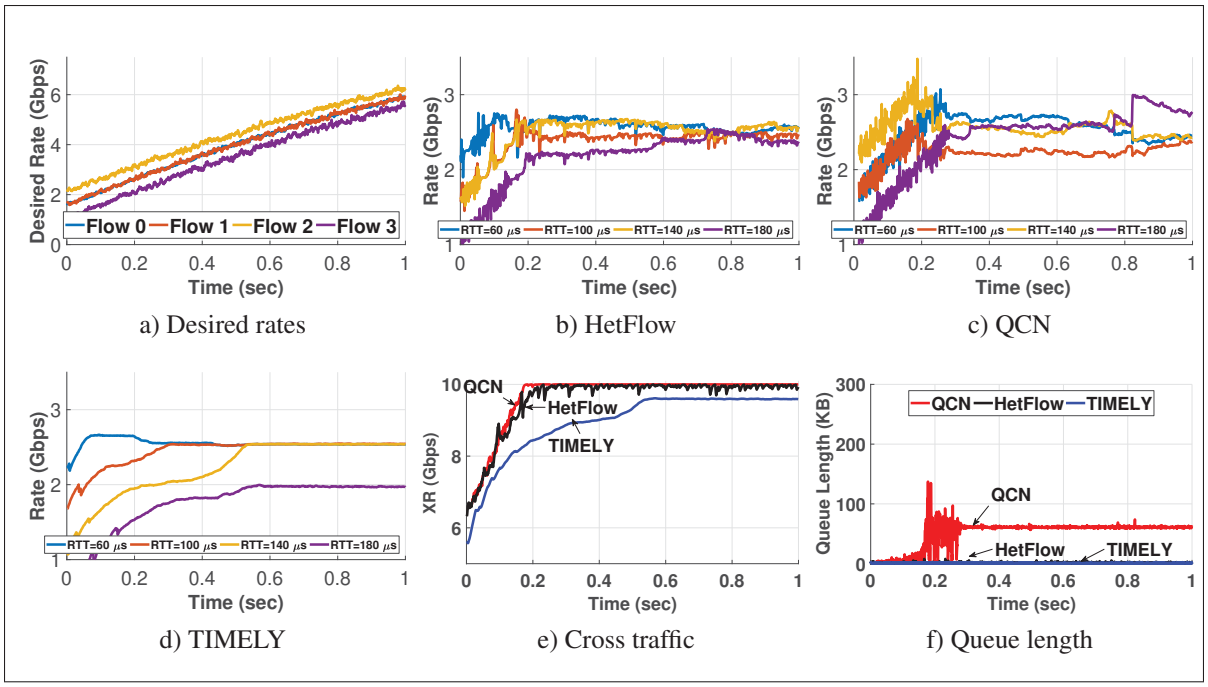


Figure 3.12 Simulation results (Fairness between flows of different RTTs)

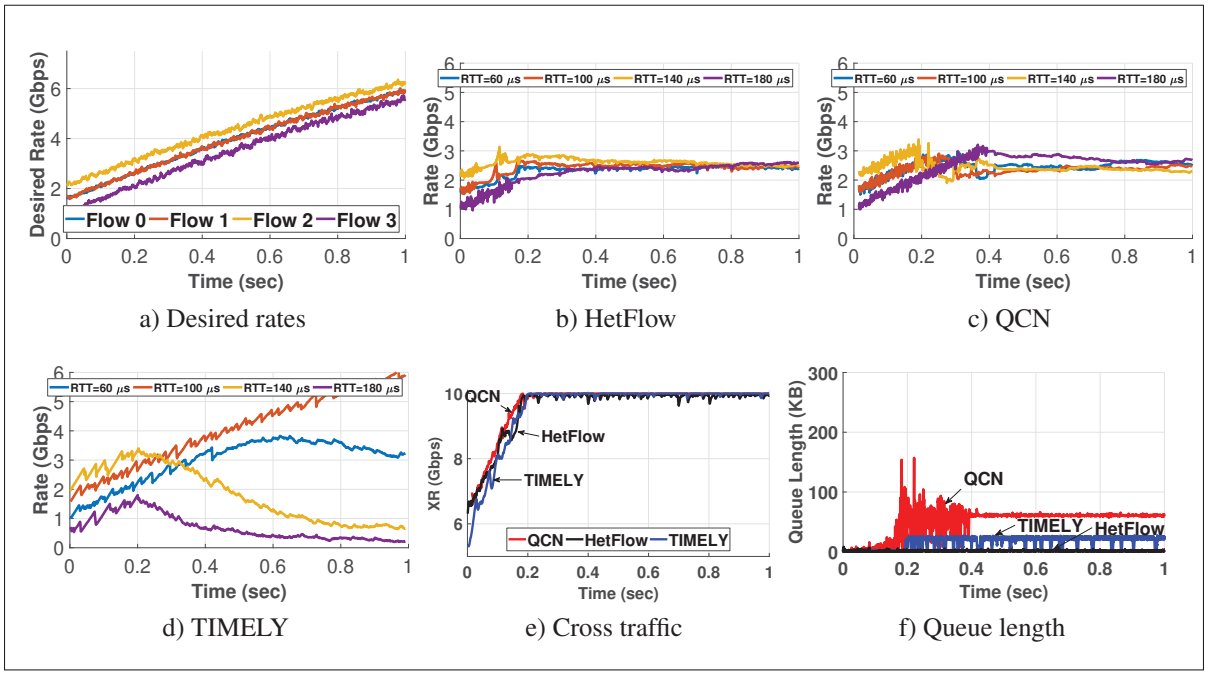


Figure 3.13 Simulation results (Fairness between flows of different packet-sizes)

issue. HetFlow derives the fairness between flows by eliminating the dependency on RTT in both the rate increase and rate decrease processes represented by 3.6 and 3.5 respectively. In this section, we conduct several simulations to evaluate that claim.

3.7.1 Experiment I - Fairness Between Flows of Different RTTs

In this experiment, we investigate the fairness between flows of different RTTs using the same simulation topology shown in Fig. 3.8. In this simulation, all flows send using the desired rate shown in Fig. 3.12a. Flow 0, 1, 2, and 3 experience RTTs equal 60, 100, 140, and 180 μs respectively. Fig. 3.12b shows that HetFlow achieves better fairness between flows compared to QCN and TIMELY as shown in Fig. 3.12c and Fig. 3.12d respectively. This behavior matches our expectation and confirms that HetFlow is not affected by RTT. In contrast, the RTT-based reaction of TIMELY forces TIMELY to achieve multiple stable points as shown in Fig. 3.12d. Fig. 3.12e depicts that both HetFlow and QCN succeeded in keeping cross traffic XR limited to the maximum link capacity. However, TIMELY, response time is larger and requires a longer time to saturate the link. Bandwidth recovery could be accelerated by increasing δ which could reduce queue stability and causes queue fluctuation. In addition, Fig. 3.12f shows that QCN experiences queue fluctuations while HetFlow and TIMELY kept the queue length close-to-zero.

3.7.2 Experiment II - Fairness Between Flows of Different Packet-sizes

In this experiment, we investigate the fairness between flows of different packet sizes using the simulation topology shown in Fig. 3.8. In this simulation, we consider 4 flows (Flow 0, 1, 2, and 3) that send with desired rates as shown in Fig. 3.13a. Flows 0, 1, 2, and 3 send data with average packet sizes equal to 200, 600, 1000, 1400 Bytes respectively. Fig. 3.13b shows that HetFlow succeeded in achieving better fairness between flows compared to QCN (Fig. 3.13c). TIMELY subtracts the serialization time while calculating RTT which forces RTT to represent network delay purely. However, this assumption is valid only for cut-through switches and when data frames do not experience queuing through the data-path. Therefore,

in real networks when data frames get queued, this serialization time is accumulated at each switch. Therefore, this assumption becomes no longer valid. In addition, store-and-forward switches add this serialization time at each hop, which invalidates this assumption too. Since the serialization time depends on the packet size, it is expected that TIMELY will react poorly when packet sizes vary which is clearly shown in Fig. 3.13d.

Fig. 3.13e reveals that HetFlow, QCN and TIMELY prevented congestion by keeping the cross traffic limited to the maximum link capacity. Nevertheless, HetFlow achieves a better queue stability as shown in Fig. 3.13f. In fact, HetFlow kept the queue length close-to-zero while QCN suffered from queue fluctuations and TIMELY has a higher queue length.

3.8 Testbed Implementation

In this section, we present a testbed implementation to validate the performance of HetFlow. HetFlow testbed is implemented using DPDK. The testbed contains 3 Linux hosts that are connected to a 10-Gbps switch (Fig. 3.14). In our implementation, HetFlow parameters are set as shown in Table 3.2. In addition, we collect testbed statistics using a third host in order to plot transmission rates R, and cross traffic rate XR.

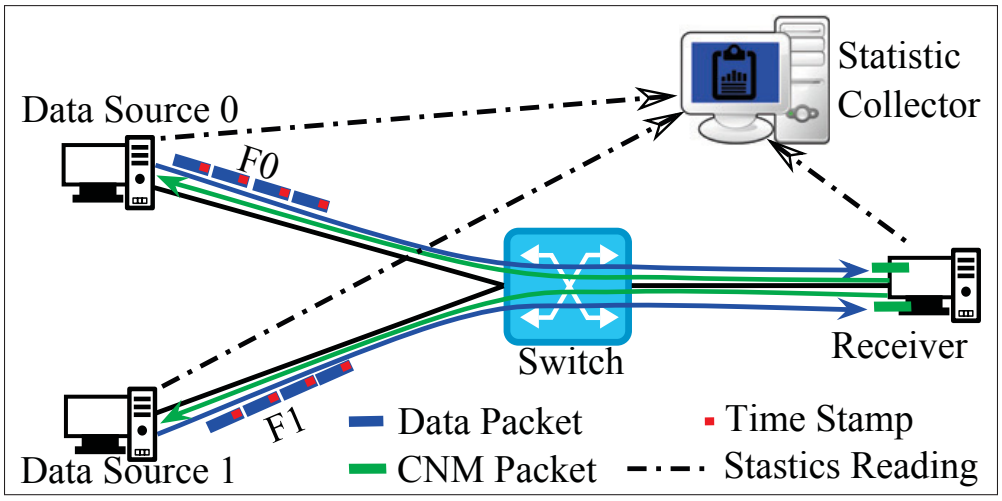
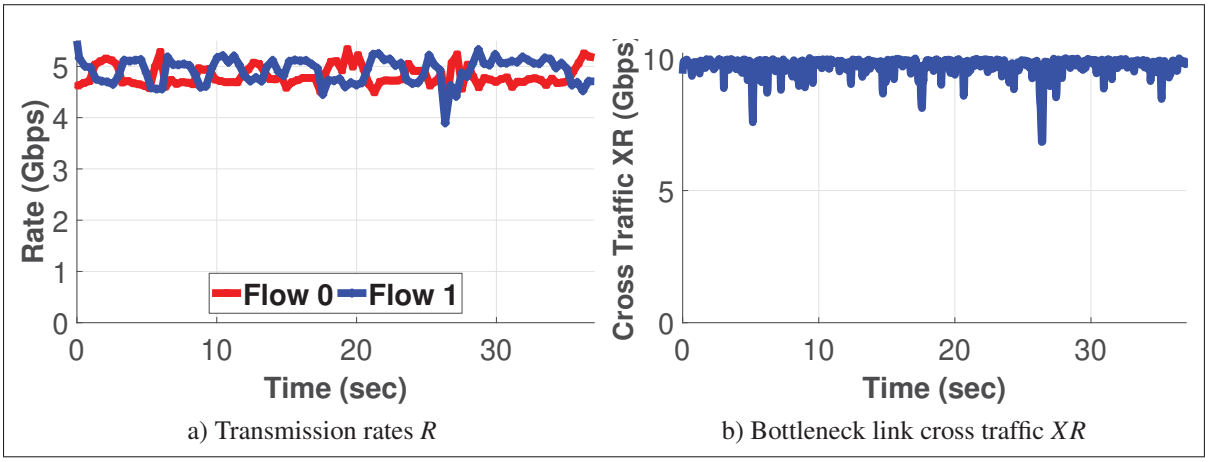


Figure 3.14 Testbed network

HetFlow rate controller is implemented in DPDK by injecting a pacing delay between packets. In this implementation, we calculate the transmission time t_{tr_next} of the next packet according to (3.23):

$$t_{tr\_next} = \frac{L}{R} + t_{tr}. \tag{3.23}$$

Where L is the last served packet length and t_{tr} is its transmission time. Thus, when a new packet arrives, HetFlow delays this packet until t_{tr_next} passes before forwarding it. Otherwise, HetFlow sends this packet instantly. In this experiment, all data sources use UDP. Data source 0 and source 1 generate flow 0 and flow 1 respectively. Both flow 0 and 1 have a desired rate = 6 Gbps, and fixed frame size = 1500 Byte. Fig. 3.15a shows that HetFlow succeeded in preventing congestion by controlling the transmission rates of both flow 0 and 1. Fig. 3.15b depicts that HetFlow succeeded in limiting the cross traffic XR to the maximum link capacity.



a) Transmission rates R b) Bottleneck link cross traffic XR

Figure 3.15 Testbed results

Fig. 3.15a and Fig.3.15b reveal that the HetFlow succeeded in controlling hosts' transmission rates to prevent packet loss.

3.9 Summary

The emergence of Ethernet-based applications such as DCN, VXLAN, and RoCE raise the need for robust and scalable transport network. In this paper, we present HetFlow as a delay-

based Ethernet congestion control that achieves close-to-zero queue length. In addition, we present a comparison of HetFlow with an Ethernet congestion control mechanism (namely QCN) and a delay-based congestion control mechanism (namely TIMELY). Based on the simulation and the experimental results we can summarize the differences between HetFlow, QCN, and TIMELY in Table 3.3.

Table 3.3 Comparison between HetFlow, QCN and TIMELY

	HetFlow	TIMELY	QCN
Congestion metric	Delay	Delay	Queue length
Reaction Time	Fast	Fast	Medium
Queue length	Close-to-zero	Low	Medium
Rate Fluctuation	Stable	Fluctuate	Fluctuate
Fairness (different RTTs)	Fair	Not fair	Medium
Fairness (packet sizes)	Fair	Not fair	Medium
Scalability	High	Medium	Medium

HetFlow outperforms QCN and TIMELY in terms of reaction time, queue length, rate fluctuation, and fairness. One can notice that due to the probabilistic behavior of QCN, it requires several iterations of Fb calculation to achieve fairness. Therefore, QCN faces slow reaction time and fairness issues. Also, TIMELY achieves a lower queue length compared to QCN because TIMELY tries to keep a bounded delay between T_{low} and T_{hig}. In addition, TIMELY builds its response on RTT which causes fairness issue when flows of different RTTs share the same data path. Further, because the serialization delay depends on frame size, TIMELY also faces fairness issue when flows of different packet sizes share a data path.

Implementation remarks: HetFlow implementation requires addressing the following remarks. First, when HetFlow-capable flows share the same data path with non-HetFlow-capable flows, it is most likely that the latter will monopolize the bandwidth. This issue can be eas-

ily solved by isolating HetFlow-capable flows in a separate priority class using IEEE 802.1p Ethernet classification.

Second, even though HetFlow uses OWD to detect congestion, no clock synchronization between hosts is required. As HetFlow uses OWD variation, any clock shift will be canceled when included in both parts of (3.3).

Third, we ought to mention the issue of clock drift between hosts. Based on our testbed implementation, the measured clock drift is around $10\mu s$ per second. For a 10-Gbps link, the time between two consecutive samples $\eta = 8e^{-5}s$. Thus, the clock drift per sample will be around $(10e-6 \times 8e-5 = 80e^{-11}s = 0.8ns)$ which is negligible compared to an extra delay of serializing a 1500-byte packet on a 10-Gbps link ($1.2\mu s$).

Finally, both HetFlow and TIMELY adopt a derivative approach to control congestion. In contrast, a better approach would be a proportional-derivative control loop. However, using the delay as a congestion metric is not reliable because once the packet is enqueued, a serialization delay is added and can't be encountered for in the equations. Therefore, using the queue length and the delay variation would present a better proportional-derivative control system. In order to keep our commitment to not changing network switches, we consider using ECN marking as the proportional part of the control system. In addition to delay variation as the derivative part. Hence, we leave this approach for future work.

3.10 Related Work

Vast amount of research is done to reduce queuing delay in both transport and Ethernet layers in data center networks. Here, we cover few closely related ideas that we have not discussed elsewhere in the paper. In order to avoid the loss-based behavior of TCP, DCTCP is presented in (Alizadeh *et al.*, 2010). DCTCP uses ECN marking to detect congestion and requests flows to slow down.

DCQCN (Zhu *et al.*, 2015) overcomes the need of switch modification in DCQCN by using ECN marking similar to DCTCP. DCQCN tries to achieve QCN-like behavior while using the Explicit Congestion Notification (ECN) marking feature that is available in ECN-aware switches. Using ECN-marking as a congestion metric introduces limited information compared to the QCN feedback parameter. Therefore, in this paper we compared our proposal with the standard QCN protocol. However, we believe it would be interesting to compare HetFlow and DCQCN in future work.

Trading a little bandwidth in order to achieve low queue length and low latency is discussed in a number of papers. For example, HULL (High-bandwidth Ultra-Low Latency) is presented in (Alizadeh *et al.*, 2012) to reduce average and tail latencies in data center network by sacrificing a small amount of bandwidth (e.g., 10%). HULL presents the Phantom Queue (PQ) as a new marking algorithm based on link utilization rather than queue occupancy (by setting ECN bit). The challenges of HULL are the need of switch modification.

Few centralized solutions are proposed in the literature. For example, Fastpass (Perry *et al.*, 2014) embraces central control for every packet transmission which raises a scalability issue.

3.11 Conclusion and Future Work

In this paper, we presented HetFlow as a congestion control that aims to avoid congestion in Ethernet layer. HetFlow is a delay-based congestion control mechanism that exploits the delay information to detect congestion. A stability analysis and scalability study of HetFlow is presented and evaluated by simulations. Moreover, The overall feasibility and performance of HetFlow are assessed and evaluated by simulation and testbed implementations. Our results confirm that HetFlow mechanism outperforms QCN and TIMELY. It depicts that HetFlow succeeds in preventing frame loss in Ethernet network, keeps switch queue length close to zero and, consequently, reduces network latency to the minimum. In addition, HetFlow achieves fairness even between flows of different packet sizes or different RTTs.

For future work, studying a proportional-derivative congestion control mechanism based ECN marking, as the proportional part, and delay variation, as the derivative part, is considered for investigation. In addition, comparing the performance between HetFlow and ECN-based protocol is ongoing.

CHAPTER 4

ON USING CPRI OVER ETHERNET IN 5G FRONTHAUL: A SCHEDULING SOLUTION

Mahmoud Bahnasy[1], Halima Elbiaze[2]

[1] Département de Génie électrique, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Département d'informatique, Université du Québec à Montréal

This article was submitted at «IEEE/ACM Transactions on Networking » in December 2017.

4.1 Abstract

CPRI is currently the most widely used protocol for fronthaul transport between the REC and the RE. However, CPRI has very stringent requirements regarding delay and jitter. Traditionally, these requirements are met using point-to-point fiber optics which increases both CAPEX and OPEX of mobile networks. Using Ethernet as a transport network for fronthaul draws significant attention of both academia and industry. The Ethernet-based fronthaul network provides several advantages such as i) low-cost equipment, ii) sharing existing infrastructure, as well as iii) the ease of OAM.

In this paper, we introduce DTSCoE as a distributed scheduling algorithm for IEEE 802.1Qbv to support CPRI traffic over Ethernet. The results obtained through simulation shows that integrating DTSCoE and Ethernet provides a feasible solution to support CPRI traffic to achieve minimum network latency and zero jitter.

4.2 Introduction

The exponential increase in mobile network users and the enormous bandwidth required by new mobile applications lead to massive increase in mobile data traffic. It is anticipated that by 2021 smartphone subscriptions will double to 6.4 billion subscriptions exchanging 1.6 Zettabytes of

data (Ericsson, 2016). These characteristics require the envisioned 5G mobile networks to provide very high rates (up to 10 Gbps per user) and sub-milliseconds latency, particularly for time-critical applications. To achieve ultra-high user data rates, 5G networks require higher transmission frequencies which lead to shorter radio transmission distance. This could be achieved by distributing the RRHs into smaller cells.

A promising approach to reconcile these requirements, with conservative investments, is to split the mobile network node into REC (i.e. a BBU which processes baseband signals and is located in a central office) and the RE (i.e. RRHs that are distributed in each cell and consist of an antenna and basic radio functionality).

Originally, this solution was called C-RAN since many lightweight RRHs are deployed in smaller cells and connected to fewer BBUs in a centralized BBU pool. The emergence of virtualization and cloud computing with its cost efficiency, high performance, scalability, and accessibility led to a novel approach that virtualizes the BBU pool in the cloud. Therefore, the solution name changed from centralized RAN to cloud RAN C-RAN (Mobile, 2013). Moreover, an analysis on statistical multiplexing gain is performed in (Namba *et al.*, 2012). The analysis shows that in Tokyo metropolitan area, the number of BBUs can be reduced by 75% compared to the traditional RAN architecture. Further, virtualized RECs can move across different REC pools according to traffic/load requirements. Tidal effect is an example that shows the advantages of this virtualized proposal. Base stations are often dimensioned for busy hours, and users move between cells. Thus, in a given period when users move, for example, from office to residential areas, a huge amount of processing power is wasted in the regions where the users have left. By moving the digital processing units into a centralized location, network resources (in this case a BBU pool) could be allocated/deallocate based on traffic load. Consequently, it increases network efficiency and reduces cost.

In C-RAN, the separation between REC and RE introduces the Fronthaul network as shown in Fig. 4.1. This fronthaul network is responsible for carrying digitized complex I/Q radio samples between the RRHs and the BBUs.

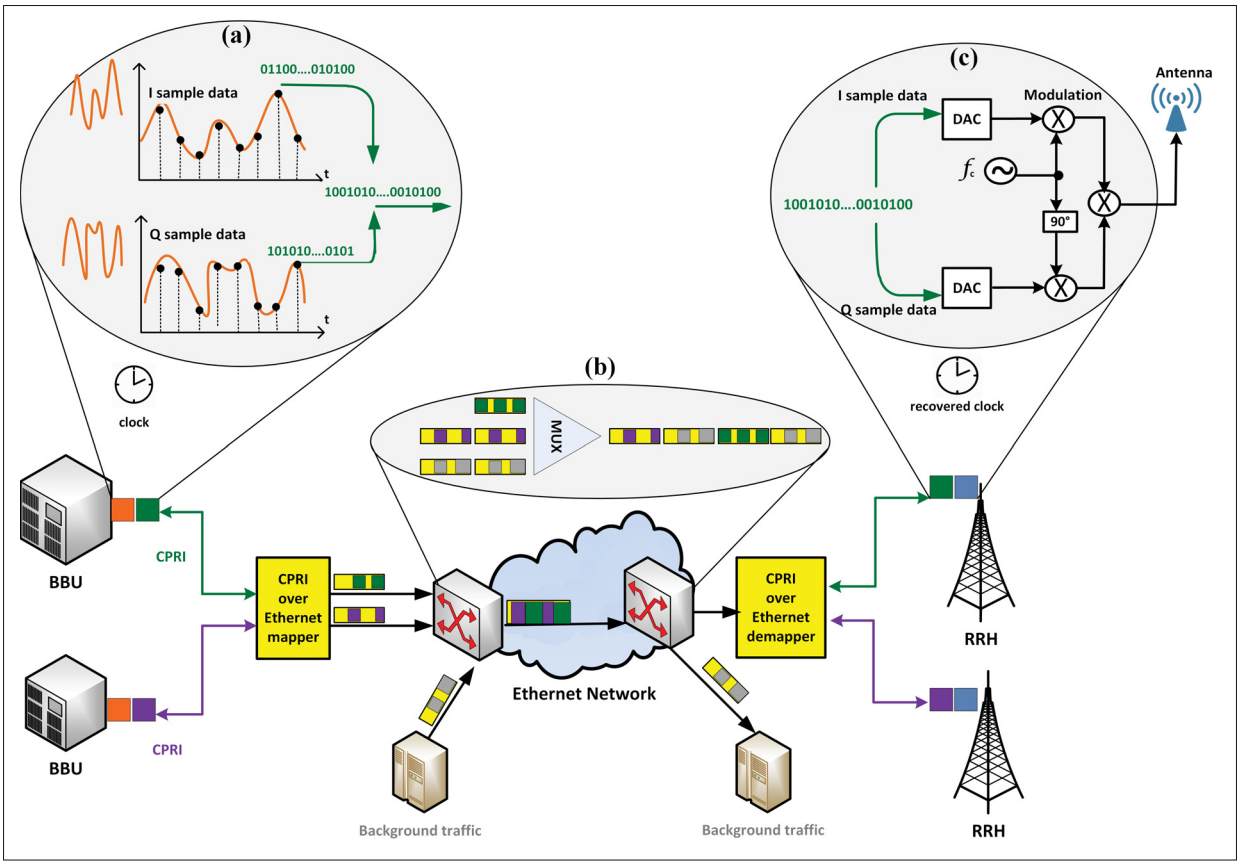


Figure 4.1 CPRI over Ethernet overview

Several ongoing projects, such as Time-Sensitive Networking for Fronthaul IEEE 802.1CM (Institute of Electrical & Electronic Engineers, 2017a), Packet-based Fronthaul Transport Networks IEEE P1914.1 (Institute of Electrical & Electronic Engineers, 2017b) and RoE Encapsulations and Mappings IEEE P1914.3 (Institute of Electrical & Electronic Engineers, 2017c) strive to define an interface between REC and RE. CPRI (Ericsson AB, Huawei Technologies Co. Ltd, NEC Corporation, Alcatel Lucent, and Nokia Networks, 2015) is defined as the internal interface between REC and RE. CPRI is designed based on the digital radio over optical fiber concept where the radio signal is sampled, quantized and transmitted over optical network. However, optical networks could be cost inefficient in some scenarios; e.g. building an optical network to connect RRHs, that are distributed in a skyscraper floors, is cost-inefficient. Whereas, building an Ethernet network or using existing networks introduces huge cost reduc-

tion. Therefore, a cost-efficient, flexible and re-configurable mobile fronthaul that supports emerging network paradigms becomes imperative.

Transporting CPRI over Ethernet network has recently drawn the attention of both the industry and the academia because of its cost efficiency. Ethernet network is widely used in access and data-center networks. It has also shown huge capacity growth lately. Accordingly, encapsulating CPRI traffic over Ethernet introduces significant savings in CAPEX and OPEX. Furthermore, the OAM capabilities of Ethernet provide standard methods for network management and performance monitoring. However, CPRI traffic has very stringent requirements in terms of jitter, latency, bandwidth, bit error rate BER, and network synchronization that must be satisfied by the transport network.

4.3 CPRI: Challenges and Requirements

CPRI is a serial line interface that transmits isochronous data frames in the fronthaul network; i.e. CPRI flows have regular inter-frame intervals. CPRI provides the physical and the data link layer to transport radio information between REC and RE. Fig. 4.1 presents the downlink processing for CPRI. The REC generates the radio signal, samples, quantizes, and sends it to the RE (Fig. 4.1a). At Ethernet network edges, CPRI flows are encapsulated, multiplexed and transmitted through the network (Fig. 4.1b). The RE reconstructs the signal and transmit it over the air (Fig. 4.1c). The inverse occurs in the uplink stream. Consequentially, the RE is simplified and does not require digital processing.

Ethernet Transport Network (ETN) is used as a fronthaul service to carry CPRI traffic (Fig. 4.1b). A CPRI over Ethernet mapping module is installed in the boundary between BBUs and the Ethernet transport network. This is where the CPRI signal arrives at the ETN (for both the radio side and the baseband side), and where the CPRI signal departs the ETN (for both the radio side and the baseband side). CPRI at each baseband side runs with its original clock, which is likely asynchronous to the ETN's clock. Besides, CPRI's latency measurement

mechanism uses its local Time of Day (ToD), independent of the Ethernet ToD, and requires a symmetric uplink and downlink.

4.3.1 Delay/Jitter Requirement

CPRI interface must satisfy fronthaul network requirements in terms of delay, jitter and bandwidth. In C-RAN, the base stations are required to prepare an ACK/NACK within 3 ms including BBU processing time ($\approx 2700\mu s$) and RRH processing time ($\approx 50\mu s$), besides propagation delay, which leaves $100-200\ \mu s$ for fronthaul one-way delay (Mobile, 2013).

Fronthaul network also has a stringent requirement of several nanoseconds in terms of jitter in order to reconstruct the waveform. Traditionally, jitter requirement is addressed using buffering at egress devices. However, implementing buffering at egress devices must accommodate the worst-case delay which introduces a fixed delay that is equal to the worst-case delay. While this is suitable for data streaming, it introduces an extra delay that might degrade the performance of CPRI traffic.

4.3.2 Data Rate Requirement

Due to the high data rate requirement of 5G networks, CPRI entails stringent requirement in terms of transmission rate. CPRI transmission rate increases proportionally with the number of antennas per sector, the sampling rate and sample width (bits/sample). For example, a four-antenna site, 2×2 multiple-input and multiple-output MIMO channel of 20 MHz bandwidth, requires 9.83 Gbps (de la Oliva *et al.*, 2016). These requirements are currently met with point-to-point fiber optics. However, Ethernet networks introduce huge cost reduction compared to optical networks in some scenarios.

Moreover, CPRI compression in the fronthaul network is presented in the literature to address the high rate requirement. Once CPRI traffic is compressed, it can be encapsulated and transmitted according to one of two options (i) size-based encapsulation (fit as many samples into one frame), or (ii) time-based encapsulation (constant number of samples per packet). The first

option reduces link overhead caused by frame header. However, it can increase the queuing delay because the sender needs to wait until there is enough data to fill an entire frame which causes variable delays (jitter). Therefore, the second approach is preferred by mobile network providers because it produces traffic with steady delays (low jitter) (Valcarenghi *et al.*, 2017).

4.3.3 Frequency Synchronization and Timing Accuracy

The separation between radio channels in mobile networks must be fulfilled with a minimum frequency accuracy of ± 50 parts-per-billion (ppb) (LTE, ETSI, 2009). Consequently, time alignment error between I/Q samples (i.e. clock shift) shall not exceed 65 *ns*. This raises the need for stringent clock synchronization between REC and RE in fronthaul network.

IEEE 1588 Precision Time Protocol (PTP) is developed as part of G.8275.1 standard to provide synchronization at Layer 2 and above for different network technologies. Also, ITU-T has defined Synchronous Ethernet (Sync-E) in G.8261 standard that requires all network nodes to participate in clock synchronization. Other research such as Network Time Protocol (NTP) provides synchronization at Layer 3. Therefore, it is expected that the fronthaul network must use one of the clock synchronization protocols mentioned earlier.

4.4 Time-Sensitive Network Standards and CPRI over Ethernet

The Time-Sensitive Network (TSN) task group, in IEEE 802.1, is working on a set of standards aiming to provide synchronized, low latency, and high bandwidth services for Ethernet networks. IEEE 802.1 was not developed originally for fronthaul traffic. Therefore, Time-sensitive networking for fronthaul (IEEE 802.1CM) project was proposed as a TSN profile for Fronthaul network. TSN project consists of tools such as:

- IEEE 802.1Qbu: Frame Preemption.
- IEEE 802.1Qbv: Enhancements for Scheduled Traffic.
- IEEE 802.1Qcc: Stream Reservation Protocol (SRP).

In IEEE 802.1Qbu frame preemption, when a Time-Critical Frame (TCF) is received while transmitting Non-Time-Critical Frame (NTCF), the switch stops processing the NTCF and begins processing the TCF.

IEEE 802.1Qbv is a time-aware traffic shaper that opens or closes the output port gate for a particular traffic class. High priority traffic is transmitted as soon as its gate opens which can reduce jitter. Since IEEE 802.1Qbv does not specify any scheduling algorithm, a proper scheduling algorithm is crucial to reduce jitter on Ethernet-based fronthaul. Several simulation experiments are presented in (Wan & Ashwood-Smith, 2015) which demonstrate promising results using a simple IEEE 802.Qbv scheduling algorithm. This algorithm reserves timeslots for IEEE 802.1Qbv at each network node independently. Such an algorithm resolves contention at each network node by queuing CPRI frames which shift their timeslots without coordination with the other network nodes. This lack of coordination causes extra delay/jitter, in some cases, as shown by their results where the jitter increases up to 1000 *ns* when packet or timeslot sizes vary.

In addition, IEEE 802.1Qcc SRP is introduced as part of TSN project to be implemented on top of the existing Multiple Registration Protocol (MRP) 802.1Qak. The MRP protocol allows streams to register their attributes (e.g. bit rate, maximum delay) across the network.

In light of that, we present a solution using Ethernet technology as a fronthaul network while fulfilling the delay and jitter requirements.

4.5 Distributed Timeslot Scheduler for CPRI over Ethernet

To address CPRI requirements regarding delay and jitter, we propose DTSCoE as a distributed timeslot scheduling algorithm for IEEE 802.1Qbv. DTSCoE uses IEEE802.1cc SRP to declare and register timeslots as resources for certain flows. Our proposed solution adopts a distributed algorithm to propagate timeslot information through network path without any centralized coordination. This distributed coordination guarantees that each registered CPRI frame passes

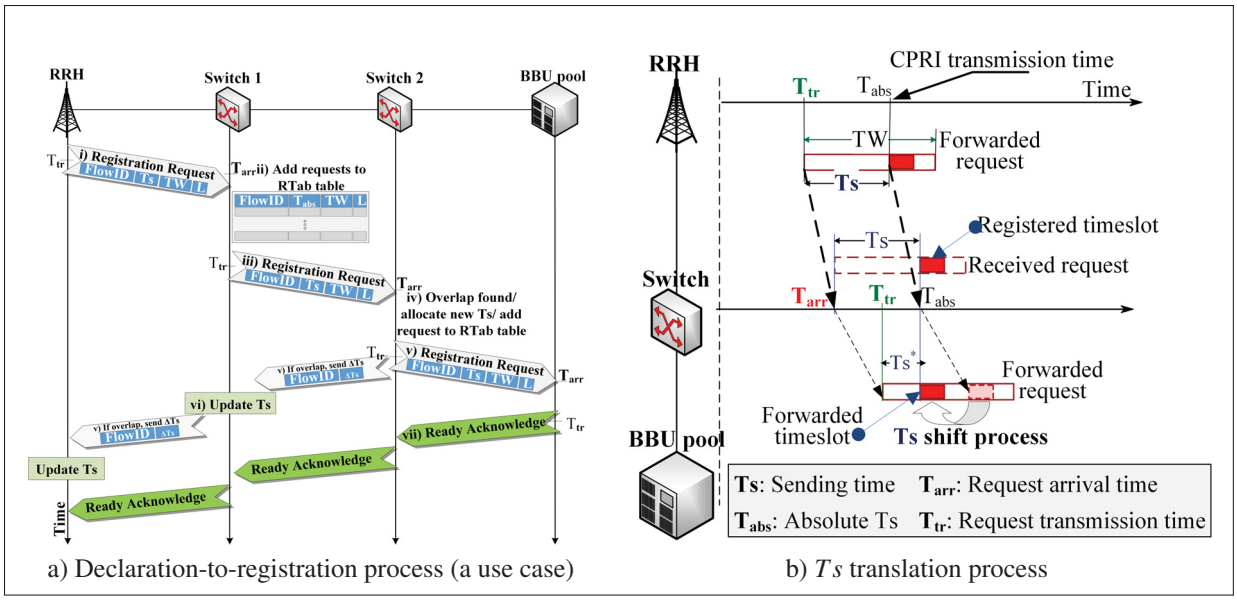


Figure 4.2 DTSCoE operations

through the network with no queuing which produces minimum latency and no jitter while avoiding the scalability issue of centralized solutions. However, such an algorithm distributes the calculation overhead among all network nodes which might cause extra overhead. Therefore, a further investigation to the computational overhead induced by DTSCoE is required.

In addition, centralized solutions produce extra complexity because it requires explicit consideration of transmission time and propagation delay (Wan *et al.*, 2016). Therefore, the centralized solution requires, in addition to the network topology, network dimensions in terms of cables' length and capacity. While in our proposal, the transmission/propagation delay is applied for reservation requests as well as CPRI frames. Therefore, no need to explicitly consider it in the calculation as explained in detail in section 4.5.3.

DTSCoE is based on a structure-agnostic encapsulation of CPRI traffic as proposed by IEEE P1914.3. In CPRI over Ethernet, time-based encapsulation is preferable where a fixed number of CPRI basic frames are encapsulated in one single Ethernet frame. Thus, we can slide the encapsulation process within a predefined transmission cycle. Hence, each CPRI source propagates its sending intention to each receiver by defining a sending time Ts within its periodic

transmission cycle, which we call time window TW. In DTSCoE, CPRI sources do not propagate a time-based timeslot, instead, they define their timeslot based on the maximum frame size L. This allows network switches to define timeslot duration t_L based on their link capacity C as follow $t_L = L/C$.

4.5.1 Declaration-to-registration

Fig. 4.2a depicts a declaration-to-registration use case that takes place by the source as follow:

i. A source (RRH in this case) sends a registration request that contains its stream attributes (in this case: flow ID, Ts, TW, and L) toward a receiver.

ii. When switch 1 receives this registration request, it adds this request in a local table called registration table *RTab* table. This table is used to define a timeslot occupation vector $TsVec$ of size equal to the least common multiple LCM of all declared TWs. Then, switch 1 searches for overlap in its local $TsVec$ vector.

iii. In this case, switch 1 does not find any overlaps, consequently, it forwards the registration request to switch 2.

iv. At switch 2, an overlap is found, therefore, switch 2 allocates a new timeslot, based on the occupancy of $TsVec$ vector, and slides Ts accordingly. All instances of the newly allocated timeslot within the LCM window must be available; i.e. all time slots that start at $Ts_i = i \times TW + Ts$ where $i = \{0, 1, \ldots, LCM/TW\}$ must be available.

v. Afterwards, Switch 2 forwards the registration request to the BBU pool and sends an update acknowledgement backwards to inform the previous network nodes about the timeslot modification $\Delta Ts = Ts_{alloc} - Ts$.

vi. Each switch/host, that receives the update acknowledgement, updates its *RTab* table and $TsVec$ vector accordingly.

vii. At the BBU pool, a ready/fail acknowledgement is created and sent back to the source.

viii. Finally, the source starts CPRI transmission at the agreed Ts once a ready acknowledge is received.

This declaration-to-registration process always occurs in a single direction from the declaring participant to its adjacent node. Thus, This process is repeated until it reaches the receiver. Additionally, each node adds its currently occupied timeslots to the forwarded request. By adding the occupied timeslots, in the registration request, each node can avoid allocating timeslots that are already allocated in other nodes, along the data path.

Because DTSCoE is a distributed mechanism, a timeslot overlaps could occur after applying ΔTs with a recently registered request. To overcome this issue, Each network node serves new registration requests once all the pending requests are fulfilled. Another approach to address this issue is to serve all requests simultaneously and send a fail/update acknowledge once overlap is detected. Therefore, the source sends a new registration request upon failure.

4.5.2 Contention Resolution

DTSCoE uses the fact that multiple CPRI basic frames are encapsulated in one single Ethernet frame and that we can slide the encapsulation process within a predefined time window. Therefore, in order to avoid long negotiation between network nodes, if the requested timeslot is occupied, we propose that each CPRI flow registers a timeslot defined by a flexible start time Ts within a time window TW. Thus, network nodes can reserve a new timeslot within the requested TW in case of overlap (step (iv) in Fig. 4.2a). We believe this relaxed constraint can reduce the registration phase to one step rather than renegotiating new timeslot.

By sliding the encapsulation process within the window size, we give the switch enough time to finish transmitting each flow frame in its dedicated timeslot. Therefore, CPRI frames are transmitted sequentially with zero queuing delay. In addition, one queue is required to support CPRI traffic of all flows, contrary to the proposed solutions in (Wan & Ashwood, 2015; Wan *et al.*, 2016) that need one queue per CPRI flow.

4.5.3 Ts Translation Process

DTSCoE requires strict clock synchronization, among network nodes, in order to perform timeslot reservation. This synchronization requirement could be achieved using protocols such as Sync-E, IEEE 1588 precision time protocol PTP or network time protocol (NTP).

However, we propose Ts translation as a simple process that aligns Ts across network nodes without the need for per-node clock synchronization. Yet, CPRI traffic requires clock synchronization between REC and RE[1]. DTSCoE uses the registration request transmission/receiving times at each node as reference times; i.e. TW start time is mapped to the transmission time T_{tr}, before transmitting the request, and to the arrival time T_{arr} once it is received as shown at the switch in Fig. 4.2b. Therefore, the absolute Ts is calculated at each network node using Equation 4.1 and forwarded Ts^* is recalculated before re-transmission using Equation 4.2.

$$T_{abs} = Ts + T_{arr} \pmod{TW} \tag{4.1}$$

$$Ts^* = (T_{abs} - T_{tr}) \pmod{TW} \tag{4.2}$$

Thus, Ts is not an absolute time, rather, it is the period between TW start time (request transmission time T_{tr}) and the transmission time of CPRI traffic (the absolute sending time T_{abs}) as shown in Fig. 4.2b at the RRH.

Performing the same process at each network node forces the request timeslot to be delayed only by the propagation/transmission delay. We add padding to the registration request to become of the same length as CPRI frames; therefore, the transmission delay becomes identical for reservation frame and CPRI frames. Hence, shifting the reservation inside TW by the processing time resolves the per-node synchronization issue while considering the propagation/transmission delay implicitly.

[1] REC-RE synchronization are out of scope of this paper

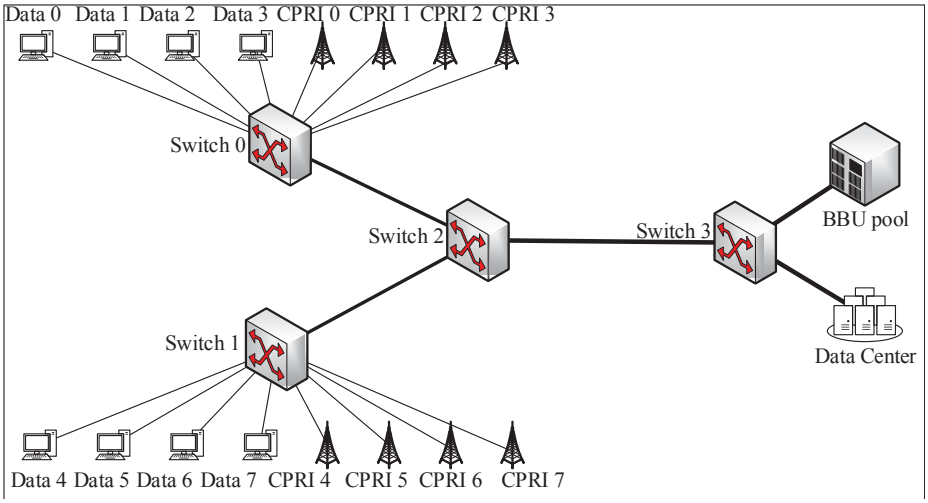
Fig. 4.2b, depicts an example of Ts translation process at one switch. It depicts that, when the switch receives a registration request at T_{arr}, it recalculates the Ts^* before transmission as depicted in Equation 4.2. Therefore, the newly calculated Ts in reference to T_{tr} matches the received Ts.

In addition to clock synchronization issue, clock drifting could cause timeslot overlap. To overcome this issue, each CPRI node updates the registration process after a predefined time T. Further, we propose using a guard band before the timeslot start time that is equal to the largest possible time drift per T.

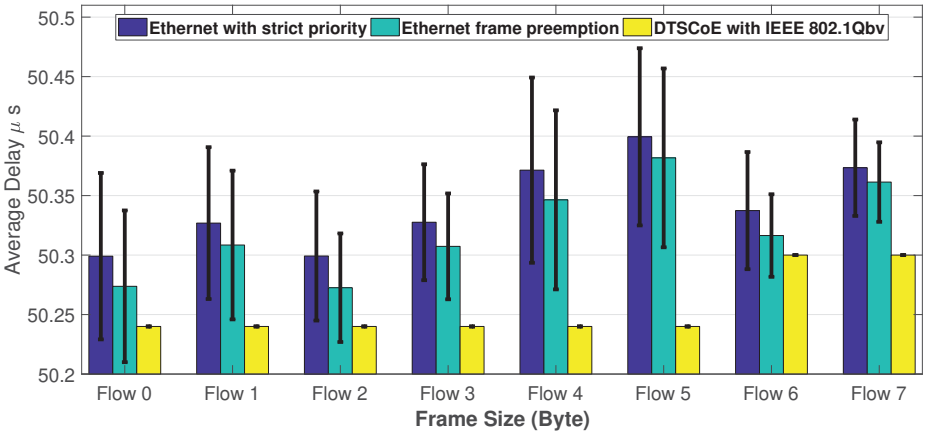
4.6 Numerical Results and Discussions

Simulation experiments are conducted to verify the performance of our proposal while multiplexing CPRI flow of different packet sizes, inter-packet gaps, and rates by setting transmission parameters of each flow as depicted in Table 4.1. In this simulation, we set two aggregation levels. The first level consists of two groups, each group consists of 4 data flows and 4 CPRI flows that are connected to one switch (Fig. 4.3a). In the second aggregation level, The output of each the two switches of the first level are connected to one switch that is connected, through another switch, to one BBU pool and one data center (Fig. 4.3a). As depicted in Table 4.1, each CPRI flow sends with a constant frame size and a fixed inter-packet gap. Data flows vary their frame sizes using normal distribution (Average = 600 Bytes, standard deviation = 150) and inter-packet gap using exponential distribution. All links in this topology have a capacity of 100 $Gbps$.

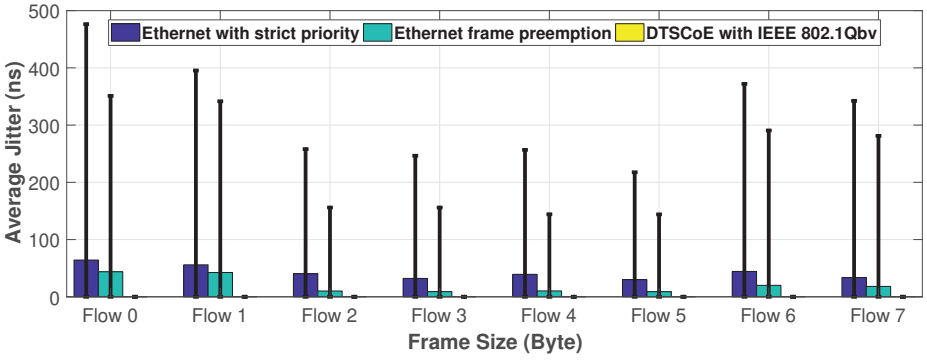
In this experiment, we compare the performance of DTSCoE against Ethernet with strict priority and Ethernet with frame preemption. Fig. 4.3b shows the average delay and its standard deviation. It shows that DTSCoE outperforms Ethernet with and without frame preemption. It shows also that DTSCoE achieves no delay variation. Moreover, Fig. 4.3c depicts the jitter and its maximum and minimum values. The results demonstrate that Ethernet without frame preemption induces jitter up to 480 ns, while Ethernet with frame preemption introduces jit-



a) Network topology



b) Delay and delay standard deviation



c) Jitter and its min/max values

Figure 4.3 DTSCoE Simulation results

Flow ID	CPRI			Data
	Frame size	Inter-packet gap	Rate	Average Rate
0	600 B	640 ns	7.5 Gbps	7.5 Gbps
1	600 B	1920 ns	2.5 Gbps	2.5 Gbps
2	750 B	960 ns	6.25 Gbps	6.25 Gbps
3	600 B	1920 ns	2.5 Gbps	2.5 Gbps
4	600 B	640 ns	7.5 Gbps	7.5 Gbps
5	600 B	1920 ns	2.5 Gbps	2.5 Gbps
6	750 B	960 ns	6.25 Gbps	6.25 Gbps
7	600 B	1920 ns	2.5 Gbps	2.5 Gbps

Table 4.1 CPRI/Data transmission parameters

ter as high as 360 *ns*. In contrast, DTSCoE achieves zero jitter regardless of frame size or inter-packet gap.

One can foresee that background traffic might experience long queuing delay due to CPRI traffic timeslot reservation. Therefore, a proper flow control mechanism must be used to control its transmission (e.g. IEEE 802.1Qbb PFC).

4.7 Conclusion

In this paper, we proposed DTSCoE as a distributed scheduling algorithm for IEEE 802.1Qbv that allows using Ethernet technology to transport CPRI traffic. In DTSCoE, sources use SRP (IEEE 802.1Qcc) to declare their intention of transmission by sending a registration request, which includes sending time Ts, time window TW and maximum frame length L, through data path. A simulation experiment is conducted to verify the performance of DTSCoE while multiplexing CPRI flow of different packet sizes, inter-packet gaps, and rates at multiple aggregation levels. The simulation results reveal that DTSCoE satisfied CPRI requirements and achieved the minimum network latency and zero jitter.

CONCLUSION AND RECOMMENDATIONS

In this research, we study the enhancements presented by DCB task group for Ethernet network. Then, we study the possibility of providing lossless Ethernet using these enhancements without modifying Ethernet switches. In this context, we propose three Ethernet applications that require lossless Ethernet namely i) switch fabric for routers, ii) lossless data center fabric, and iii) zero-jitter fronthaul network for CPRI over Ethernet for 5G network. we present ECCP as a new congestion control solution that is built on new router architecture to avoid steady-state congestion altogether. We analyzed ECCP using phase plane method while taking into consideration the propagation delay. Our stability analysis identifies the sufficient conditions for ECCP system stability. In addition, ECCP performance is verified by simulation and Linux-based implementation. The obtained results reveal that the ECCP system is stable, and it achieves close-to-zero queue length.

Moreover, we present HetFlow as a congestion control that aims to avoid congestion in data center networks. The overall feasibility and performance of HetFlow are assessed and evaluated by DPDK-based testbed implementation and simulation. Our results confirm that HetFlow mechanism outperforms QCN and TIMELY and succeeds in preventing frame loss in Ethernet network, keeps switch queue length close to zero and, consequently, reduces network latency to the minimum. A mathematical analysis is also presented in this research to study the stability of HetFlow.

Furthermore, we presented DTSCoE as a distributed scheduling algorithm for IEEE 802.1Qbv that uses IEEE 802.1Qcc. The overall feasibility and performance of DTSCoE are assessed and evaluated through simulation. Our results show that Ethernet network is capable of achieving a network of zero-jitter and minimum latency if integrated with a proper scheduling algorithm.

For future work, we propose designing a proportional-derivative congestion control mechanism based on delay variation, as the derivative part and ECN marking, as the proportional part. In

addition, the effect of our proposed solutions on different data center applications needs more investigation.

APPENDIX I

PROOF OF LEMMA 2.1 (STABILITY CONDITIONS OF THE ECCP RATE DECREASE SUBSYSTEM)

Proof. Starting with ECCP rate decrease subsystem equation (2.17) that could be presented as follows:

$$y'(t)+\frac{G_d}{T}\Big(y(t-\tau)+w\times T\times y'(t-T-\tau)\Big)(y(t)+\zeta)=0. \tag{A I-1}$$

Lyapunov has shown that the stability of nonlinear differential equations in the neighborhood of equilibrium point can be found from their linear version around the equilibrium point (Arnold, 1978) when the Lipschitz condition is satisfied. For delay differential equations (Driver, 2012) has proven that, delay differential equations is uniformly asymptotic stable if its linearized version is uniformly asymptotic stable and the Lipschitz condition is satisfied.

Consider functions g_1 and g_2 are defined as follow: $g_1(t)=y(t)$ and $g_2(t)=-\frac{G_d}{T}(y(t-\tau)+wTy'(t-T-\tau))(y(t)+\zeta)$. Since both $g_1(t)$ and $g_2(t)$ are polynomials, for any $\vec{z}=(z_1,z_2)=(y(t),y'(t))$, there exists L such that $||g_i(t,\vec{z}_1)-g_i(t,\vec{z}_2)||\leq L||\vec{z}_1-\vec{z}_2||$, where i = 1,2. Then the Lipschitz condition is satisfied. Hence, the stability of the delay differential equation is defined by the stability of the linearized part near the equilibrium point.

Thus, the linear part of the rate decrease subsystem equation becomes:

$$y'(t)+\frac{G_d\zeta}{T}\big(y(t-\tau)+w\times T\times y'(t-T-\tau)\big)=0. \tag{A I-2}$$

We use Taylor series to approximate (A I-2) by substituting $y(t-\tau)$ and $y(t-T-\tau)$ using (A I-3) and (A I-4) respectively.

$$y(t-\tau)\approx y(t)-\tau y'(t)+\frac{\tau^2}{2}y''(t). \tag{A I-3}$$

$$y'(t-T-\tau) \approx y'(t) - (T+\tau)y''(t). \tag{A I-4}$$

Hence (A I-2) becomes:

$$\begin{aligned} &y'(t) + \frac{G_d\zeta}{T}\big(y(t) - \tau y'(t) + \frac{\tau^2}{2}y''(t)\big) + w\big(y'(t) - (T+\tau)y''(t)\big) \approx 0 \\ &(\frac{\tau^2}{2T} - w(T+\tau))y''(t) + \Big(w + \frac{1}{G_d\zeta} - \frac{\tau}{T}\Big)y'(t) + \frac{1}{T}y(t) \approx 0 \end{aligned} \tag{A I-5}$$

where $G_d\zeta \neq 0$. Therefore, one can derive the characteristic equation of (A I-5) as:

$$\Big(\frac{\tau^2}{2T} - w(T+\tau)\Big)\lambda^2 + \Big(w + \frac{1}{G_d\zeta} - \frac{\tau}{T}\Big)\lambda + \frac{1}{T} = 0. \tag{A I-6}$$

By calculating the roots $\lambda_{1,2}$ of the characteristic equation (A I-6), we obtain:

$$\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{A I-7}$$

where $a = \frac{\tau^2}{2T} - w(T+\tau)$, $b = w + \frac{1}{G_d\zeta} - \frac{\tau}{T}$, and $c = \frac{1}{T}$.

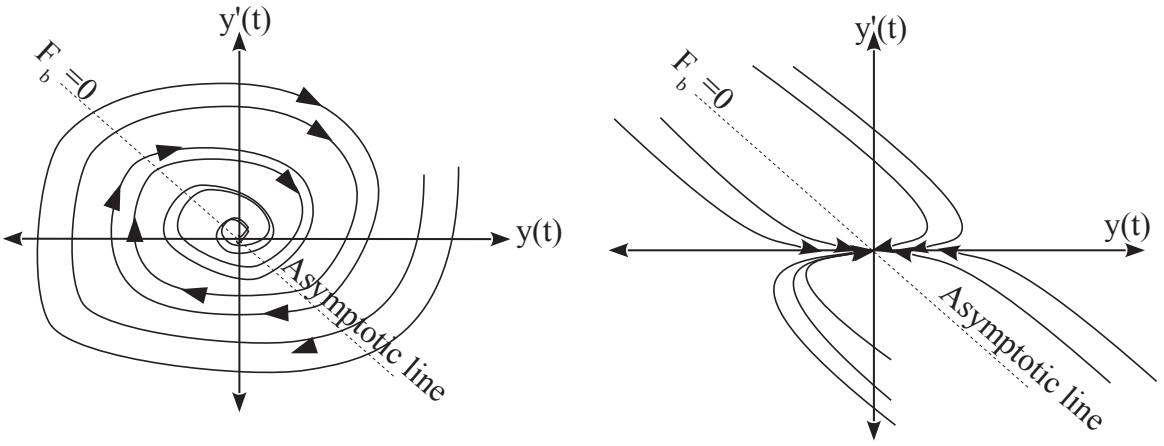
In order to study the stability of ECCP rate decrease subsystem, the roots of (A I-6) must be either (i) complex roots with negative real part for a system with stable spiral point (Fig. I-1a) or (ii) negative roots for a system with stable point (Fig. I-1b).

1. System stability with a spiral point

For a stable system with a spiral point, $\lambda_{1,2}$ must be complex numbers with negative real parts. Thus, inequalities (A I-8) and (A I-9) must hold.

$$b^2 < 4ac \tag{A I-8}$$

$$b/a > 0 \tag{A I-9}$$



(a) Complex roots with with negative real part (b) Negative real roots

Figure-A I-1 Phase trajectories of the rate decrease subsystem

Substituting a, b and c in (A I-8), we get:

$$\Big(w+\frac{1}{G_d\zeta}-\frac{\tau}{T}\Big)^2 < 4\Big(\frac{\tau^2}{2T}-w(T+\tau)\Big)\frac{1}{T}.$$

Let $H = w+\frac{1}{G_d\zeta}$, we get:

$$\Big(H-\frac{\tau}{T}\Big)^2 < \Big(2(\frac{\tau}{T})^2-4w-4w\frac{\tau}{T}+\Big)$$
$$(\frac{\tau}{T})^2+(2H-4w)\frac{\tau}{T}-H^2-4w>0. \qquad \text{(A I-10)}$$

One can say that the right-hand side (RHS) of (A I-10) represents a convex function ($\frac{d^2(RHS)}{d(\tau/T)^2} = 1 > 0$) as shown in Fig. I-2. Thus, inequality (A I-8) holds when $RHS < 0$ where $\tau/T <$

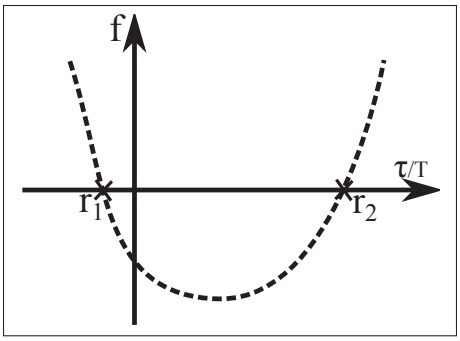


Figure-A I-2 Roots $r_{1,2}$ of a convex function

$\min(roots)$ and $\tau/T > \max(roots)$. Hence, we calculate the roots $r_{1,2}$ of the RHS as:

$$r_{1,2} = \frac{-2H + 4w \pm \sqrt{(2H-4w)^2 + 4(H^2+4w)}}{2}$$
$$r_{1,2} = -H + 2w \pm \sqrt{(H-2w)^2 + (H^2+4w)}$$
$$r_{1,2} = 2w - H \pm \sqrt{H^2 - 4wH + 4w^2 + H^2 + 4w}$$
$$r_{1,2} = w - \frac{1}{G_d\zeta} \pm \sqrt{2H^2 - 4wH + 4w^2 + 4w}. \quad \text{(A I-11)}$$

By substituting with the value of H, we get:

$$r_{1,2} = w - \frac{1}{G_d\zeta} \pm \sqrt{2w^2 - \frac{2}{(G_d\zeta)^2} + 4w}. \quad \text{(A I-12)}$$

Thus, inequality (A I-8) holds when:

$$\begin{cases} \frac{\tau}{T} < w - \frac{1}{G_d\zeta} - \sqrt{2w^2 - \frac{2}{(G_d\zeta)^2} + 4w} \\ \frac{\tau}{T} > w - \frac{1}{G_d\zeta} + \sqrt{2w^2 - \frac{2}{(G_d\zeta)^2} + 4w}. \end{cases} \quad \text{(A I-13)}$$

One can conclude that inequality (A I-8) does not hold because τ/T by definition must be limited by a certain value k ($\tau/T < k$, where $k \in \mathbb{R}^+$). Therefore, (A I-6) does not have complex roots and ECCP rate decrease subsystem does not have a stable spiral point.

2. System stability with a stable point

For a stable system with a stable point, $\lambda_{1,2}$ must be negative real number where inequalities (A I-14) and (A I-15) hold.

$$b^2 > 4ac. \tag{A I-14}$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} < 0. \tag{A I-15}$$

By developing A I-14 similar to (A I-8), we get:

$$\begin{aligned} w - \frac{1}{G_d\zeta} - \sqrt{2w^2 - \frac{2}{(G_d\zeta)^2} + 4w} &> \tau/T \\ &< w - \frac{1}{G_d\zeta} + \sqrt{2w^2 - \frac{2}{(G_d\zeta)^2} + 4w}. \end{aligned} \tag{A I-16}$$

As τ and T are always greater than zero, we can consider the positive root only.

$$\tau/T < w - \frac{1}{G_d\zeta} + \sqrt{2w^2 - \frac{2}{(G_d\zeta)^2} + 4w}. \tag{A I-17}$$

The second condition (Inequality A I-15) can be simplified as follows:

$$\frac{-b}{2a}(1 \pm \sqrt{1 - 4ac/b2}) < 0. \tag{A I-18}$$

This condition holds in one of the following two states: (i)

$$\begin{cases} -b/a > 0 \\ 1 \pm \sqrt{1 - 4ac/b^2} < 0. \end{cases} \tag{A I-19}$$

Or:

$$\begin{cases} -b/a < 0 \\ 1 \pm \sqrt{1 - 4ac/b^2} > 0. \end{cases} \tag{A I-20}$$

The second part of the first state (A I-19) does not hold in its worst case when we consider the positive root; i.e $1 + \sqrt{1 - 4ac/b^2}$ will never be less than zero. Thus we consider only the

second state. The worst case of the second part of the second state equation (A I-20) could be derived as follows:

$$\begin{aligned} 1-\sqrt{1-4ac/b^2} &> 0 \\ -\sqrt{1-4ac/b^2} &> -1 \\ 1-4ac/b^2 &> 1 \\ -4ac/b^2 &> 0. \end{aligned} \tag{A I-21}$$

For all $b^2 > 0$ and $c = 1/T > 0$, we conclude that c must be greater than 0. Consequently, b must be greater than zero (first part of A I-20) when $-a$ is greater than zero.

Hence, to satisfy the second inequality A I-20, these conditions must hold:

$$\begin{aligned} -a &> 0 \\ wT + w\tau - \frac{\tau^2}{2T} &> 0 \\ -1/2(\frac{\tau}{T})^2 + w\frac{\tau}{T} + w &> 0, \end{aligned} \tag{A I-22}$$

and

$$\begin{aligned} b &> 0 \\ w + \frac{1}{G_d\zeta} - \frac{\tau}{T} &> 0 \\ \frac{\tau}{T} &< w + \frac{1}{G_d\zeta}. \end{aligned} \tag{A I-23}$$

Dissimilar to inequality (A I-10), the RHS of (A I-22) represents a concave function. Thus, inequality (A I-22) holds when $\min(r_{1,2}) < \frac{\tau}{T} < \max(r_{1,2})$, where the roots $r_{1,2}$ of (A I-22) are:

$$r_{1,2} = w \pm \sqrt{w^2 + 2w}. \tag{A I-24}$$

Hence, inequality (A I-22) holds when:

$$w - \sqrt{w^2 + 2w} < \tau/T < w + \sqrt{w^2 + 2w}. \quad \text{(A I-25)}$$

Because τ and $T \in \mathbb{R}^+$, we consider only the positive root, thus (A I-25) becomes:

$$\tau/T < w + \sqrt{w^2 + 2w}. \quad \text{(A I-26)}$$

To conclude, ECCP rate decrease subsystem is stable with a stable point (Fig. I-1b) when inequalities (A I-17), (A I-23) and (A I-26) hold.

In summary, ECCP rate decrease subsystem is asymptotically stable and the phase trajectories of the rate decrease differential equation (2.16) are parabolas moving toward a stable node as shown in Fig. I-1b when $\tau/T < min\left(w + \frac{1}{G_d\zeta},\ w + \sqrt{w^2 + 2w}, w - \frac{1}{G_d\zeta} + \sqrt{2H^2 - 4wH + 4w^2 + w}\right)$.

□

APPENDIX II

STABILITY ANALYSIS OF ECCP RATE INCREASE SUBSYSTEM

In this appendix, we study the stability of the rate increase subsystem by substituting (2.14) into the self-increase part of (2.12).

$$y'(t) = \frac{M}{AvT \times C} \times \frac{TR - \frac{AvT \times C}{M}(y(t) + \zeta)}{2 \times T_{BC}}$$

$$y'(t) = \frac{M}{2 \times AvT \times C \times T_{BC}} \times TR - \frac{\zeta}{2 \times T_{BC}} - \frac{y(t)}{2 \times T_{BC}}. \qquad \text{(A II-1)}$$

Equation (A II-1) is an inhomogeneous second order ordinary differential equation (ODE) which has a characteristic equation of the form:

$$\lambda^2 + \frac{1}{2 \times T_{BC}}\lambda = K \qquad \text{(A II-2)}$$

where:

$$\begin{aligned} K &= \frac{M}{2 \times AvT \times C \times T_{BC}} TR - \frac{\zeta}{2 \times T_{BC}} \\ &= \frac{M \times TR}{2 \times AvT \times C \times T_{BC}} - \frac{\frac{1}{AvT} - A_{eq}}{2 \times T_{BC}} \\ &= \frac{1}{2 \times AvT \times C \times T_{BC}} \left(M \times TR - (1 - A_{eq} AvT)C\right). \end{aligned} \qquad \text{(A II-3)}$$

The phase trajectories of (A II-2) can be drawn using the Isoclinal method (Atherton & Siouris, 1977). Fig. II-1a and II-1b show the phase trajectories of the self-increase subsystem while ($K > 0$) and ($K < 0$) respectively. In general, ECCP reaches self-increase subsystem coming from the rate decrease subsystem when $M \times TR \geq (1 - A_{eq} AvT)C$. Consequently, $K > 0$ and the system follows the trajectory of Fig. II-1a. If ECCP enters the self-increase subsystem phase trajectory with $K < 0$, it follows the trajectory in Fig. II-1b for five cycles. Then, it increases

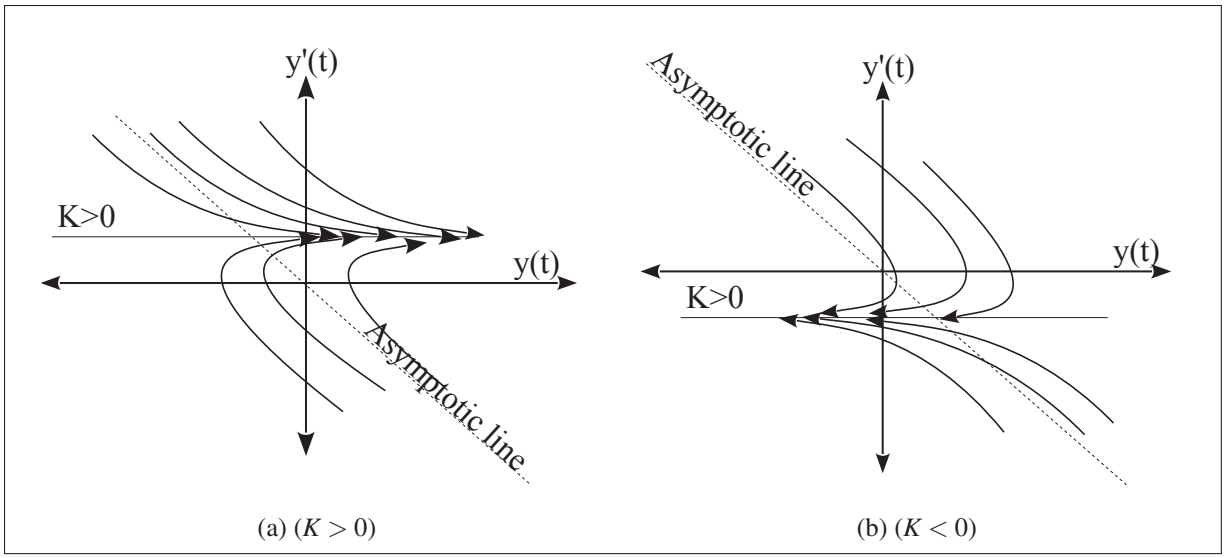


Figure-A II-1 Phase trajectories in the self-increase subsystem ($K > 0$ and $K < 0$)

TR as in equation (2.8) which increases K. Finally, ECCP follows the phase trajectory shown in Fig. II-1a.

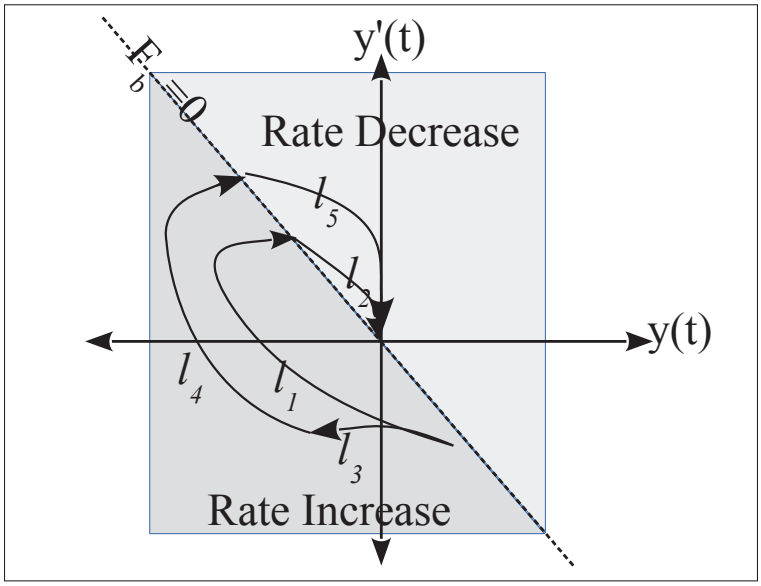


Figure-A II-2 ECCP phase trajectories

Combining Fig. I-1b, II-1a and Fig. II-1b for the rate decrease and self-increase subsystems, we get Fig. II-2. In this figure, one can notice that if the system starts in the self-increase

subsystem, it follows line l_1 ($K > 0$) toward the asymptotic line ($Fb = 0$) or it follows line l_3 ($K \leq 0$) for 5 cycles till ECCP enters AI stage and TR is increased. Then it follows l_4 toward the asymptotic line. Afterward, the system follows either line l_2 coming from FR stage to rate decrease subsystem or l_5 from the AI stage to the rate decrease subsystem. Both trajectories lead ECCP toward the equilibrium point as shown in Fig. II-2.

Therefore, ECCP rate increase subsystem is not stable, and the stability of ECCP system mainly depends on the sliding mode motion (Utkin, 1977) from self-increase subsystem into the rate decrease subsystem when the system crosses the asymptotic line ($Fb = 0$).

APPENDIX III

PROOF OF LEMMA 2.2 (BOUNDARY LIMITATIONS FOR THE ECCP)

Proof. To avoid data accumulation in the queue, the integral of the self-increase function from t to $t+(T+2\tau)$ must be less than the available bandwidth margin as depicted by (A III-1).

$$\int_{t}^{t+(T+2\tau)} MR'(t)dt < AvT\ A_{eq}C. \tag{A III-1}$$

Since ECCP is a discrete system and $R(t)$ is constant within control cycle, (A III-1) could be approximated within one control cycle to:

$$\begin{aligned} MR'(t)(T+2\tau) &< AvT\ A_{eq}C \\ M\frac{(TR-R)}{2T_{BC}}(T+2\tau) &< AvT\ A_{eq}C. \end{aligned}$$

At the equilibrium point $R=(1-AvT\ A_{eq})C/M$, and $TR>C/M$.

$$\begin{aligned} M\frac{(C/M-(1-AvT\ A_{eq})C/M)}{2T_{BC}}(T+2\tau) &< AvT\ A_{eq}C \\ \frac{(C-C-AvT\ A_{eq}C)}{2T_{BC}}(T+2\tau) &< AvT\ A_{eq}C \\ (T+2\tau) &< 2T_{BC} \\ (T+2\tau) &< \frac{2BC}{C/M} \\ BC &> \frac{C(T+2\tau)}{2M}. \end{aligned} \tag{A III-2}$$

□

In summary, ECCP keeps queue length close to zero, if condition A III-2 is satisfied.

BIBLIOGRAPHY

802.1, I. (2013). The data center bridging (DCB) Task Group (tg). Consulted at http://www.ieee802.org/1/pages/dcbridges.html.

A., J., S.V., K. R. & R., A. U. (2017). Congestion avoidance algorithm using ARIMA(2,1,1) model-based RTT estimation and RSS in heterogeneous wired-wireless networks. *Journal of network and computer applications*, 93(Supplement C), 91 - 109. doi: https://doi.org/10.1016/j.jnca.2017.05.008.

Alizadeh, M., Atikoglu, B., Kabbani, A., Lakshmikantha, A., Pan, R., Prabhakar, B. & Seaman, M. (2008, Sept). Data center transport mechanisms: Congestion control theory and ieee standardization. *Communication, control, and computing, 2008 46th annual allerton conference on*, pp. 1270-1277. doi: 10.1109/ALLERTON.2008.4797706.

Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S. & Sridharan, M. (2010). Data center TCP (DCTCP). *Sigcomm comput. commun. rev.*, 41(4), –. Consulted at http://dl.acm.org/citation.cfm?id=2043164.1851192.

Alizadeh, M., Javanmard, A. & Prabhakar, B. (2011). Analysis of dctcp: Stability, convergence, and fairness. *Proceedings of the acm sigmetrics joint international conference on measurement and modeling of computer systems*, (SIGMETRICS '11), 73–84. doi: 10.1145/1993744.1993753.

Alizadeh, M., Kabbani, A., Edsall, T., Prabhakar, B., Vahdat, A. & Yasuda, M. (2012). Less is more: Trading a little bandwidth for ultra-low latency in the data center. *Proceedings of the 9th usenix conference on networked systems design and implementation*, (NSDI'12), 19–19. Consulted at http://dl.acm.org/citation.cfm?id=2228298.2228324.

Altman, E., Barakat, C., Laborde, E., Brown, P. & Collange, D. (2000). Fairness analysis of tcp/ip. *Proceedings of the 39th ieee conference on decision and control (cat. no.00ch37187)*, 1, 61-66 vol.1. doi: 10.1109/CDC.2000.912733.

Altman, E., Jiménez, T. & Núñez-Queija, R. (2002). Analysis of two competing tcp/ip connections. *Performance evaluation*, 49(1), 43 - 55. doi: http://dx.doi.org/10.1016/S0166-5316(02)00106-2. Performance 2002.

Andrew S. Tanenbaum, D. J. W. (2011). *Computer networks, 5th edition*. Pearson.

Arnold, V. (1978). Supplementary chapters to the theory of ordinary differential equations. *Nauka, moscow*.

Association, I. T. et al. (2010). Supplement to Infiniband Architecture Specification Volume 1, Release 1.2. 1: Annex A16: RDMA over Converged Ethernet (RoCE). Apr.

Association, I. T. et al. (2014). Supplement to Infiniband Architecture Specification Volume 1, Release 1.2. 1: Annex A17: RDMA over Converged Ethernet v2 (RoCEv2). September.

Atherton, D. P. & Siouris, G. M. (1977). Nonlinear control engineering. *Ieee transactions on systems, man, and cybernetics*, 7(7), 567-568. doi: 10.1109/TSMC.1977.4309773.

Aweya, J., Ouellette, M. & Montuno, D. Y. (2001). A control theoretic approach to active queue management. *Computer networks*, 36(2–3), 203 - 235. doi: http://dx.doi.org/10.1016/S1389-1286(00)00206-1. Theme issue: Overlay Networks.

Bachmutsky, A. (2011). *System design for telecommunication gateways*. John Wiley & Sons.

Bahnasy, M., Boughzala, B., Elbiaze, H., Alleyne, B., Beliveau, A. & Padala, C. (2016, Feb). Proactive ethernet congestion control based on link utilization estimation. *2016 international conference on computing, networking and communications (icnc)*, pp. 1-6. doi: 10.1109/ICCNC.2016.7440620.

Bahnasy, M., Elbiaze, H. & Boughzala, B. (2017, May). Hetflow: A distributed delay-based congestion control for data centers to achieve ultra low queueing delay. *2017 ieee international conference on communications (icc)*, pp. 1-7. doi: 10.1109/ICC.2017.7997244.

Bahnasy, M., Elbiaze, H. & Catherine, T. (2018a, Jan). Cpri over ethernet: Towards fronthaul/backhaul multiplexing. *2018 ieee consumer communications & networking conference (ccnc)*, pp. 1-7.

Bahnasy, M. M., Beliveau, A., Alleyne, B., Boughzala, B., Padala, C., Idoudi, K. & Elbiaze, H. (2015, Aug). Using ethernet commodity switches to build a switch fabric in routers. *2015 24th international conference on computer communication and networks (icccn)*, pp. 1-8. doi: 10.1109/ICCCN.2015.7288483.

Bahnasy, M., Elbiaze, H. & Boughzala, B. (2018b). Zero-queue ethernet congestion control protocol based on available bandwidth estimation. *Journal of network and computer applications*, 105, 1 - 20. doi: https://doi.org/10.1016/j.jnca.2017.12.016.

Bailey, S. & Talpey, T. (2005). The architecture of direct data placement (DDP) and remote direct memory access (RDMA) on internet protocols. *Architecture*. Consulted at https://tools.ietf.org/html/rfc4296.

Beliveau, A., Alleyne, B., BAHNASY, M., BOUGHZALA, B., PADALA, C., ELBIAZE, H. & IDOUDI, K. (2016). Ethernet congestion control and prevention. Google Patents. WO Patent App. PCT/IB2016/050,738, Consulted at https://www.google.ca/patents/WO2016128931A1?cl=en.

Bilal, K., Khan, S. U., Zhang, L., Li, H., Hayat, K., Madani, S. A., Min-Allah, N., Wang, L., Chen, D., Iqbal, M., Xu, C.-Z. & Zomaya, A. Y. (2013). Quantitative comparisons of the state-of-the-art data center architectures. *Concurrency and computation: Practice and experience*, 25(12), 1771–1783. doi: 10.1002/cpe.2963.

Bloch, G., Crupnicoff, D., Ravid, R., Kagan, M. & Bukspan, I. (2011). Credit-based flow control for ethernet. Google Patents. US Patent App. 13/245,886.

Brakmo, L. S., O'Malley, S. W. & Peterson, L. L. (1994). Tcp vegas: New techniques for congestion detection and avoidance. *Sigcomm comput. commun. rev.*, 24(4), 24–35. doi: 10.1145/190809.190317.

Brown, P. (2000, Mar). Resource sharing of tcp connections with different round trip times. *Proceedings ieee infocom 2000. conference on computer communications. nineteenth annual joint conference of the ieee computer and communications societies (cat. no.00ch37064)*, 3, 1734-1741 vol.3. doi: 10.1109/INFCOM.2000.832573.

Charny, A., Clark, D. D. & Jain, R. (1995, Jun). Congestion control with explicit rate indication. *Communications, 1995. icc '95 seattle, 'gateway to globalization', 1995 ieee international conference on*, 3, 1954-1963 vol.3. doi: 10.1109/ICC.1995.524537.

Checko, A. (2016). Cloud Radio Access Network architecture. Towards 5G mobile networks.

Cisco Systems. (2009). Priority Flow Control: Build Reliable Layer 2 Infrastructure. Consulted at https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white_paper_c11-542809.html.

Cohen, D., Talpey, T., Kanevsky, A., Cummings, U., Krause, M., Recio, R., Crupnicoff, D., Dickman, L. & Grun, P. (2009, Aug). Remote direct memory access over the converged enhanced ethernet fabric: Evaluating the options. *2009 17th ieee symposium on high performance interconnects*, pp. 123-130. doi: 10.1109/HOTI.2009.23.

Committee, O. M. E. et al. (2012). Software-defined networking: The new norm for networks. *Onf white paper. palo alto, us: Open networking foundation*.

Community, O. (2014). OMNEST - High-Performance Simulation. Consulted at http://http://www.omnest.com/.

Croft, W. E., Eaton, L. E., Hayes, J. W. & Henderson, A. E. (2003). Transporting fibre channel over ethernet. Google Patents. US Patent App. 10/689,540.

de la Oliva, A., Hernandez, J. A., Larrabeiti, D. & Azcorra, A. (2016). An overview of the cpri specification and its application to c-ran-based lte scenarios. *IEEE Communications Magazine*, 54(2), 152-159. doi: 10.1109/MCOM.2016.7402275.

Desai, H., Snively, R. N., Vobbilisetty, S. & Wenig, G. C. (2007). Fibre channel over ethernet frame. Google Patents. US Patent App. 11/958,319.

Devera, M. (2002). Hierarchical token bucket theory. *Url: http://luxik.cdi.cz/ devik/qos/htb/-manual/theory.htm*.

Dorel, J.-L. & Gerla, M. (1997). *Performance analysis of tcp-reno and tcp-sack: The single source case*. UCLA Computer Science Department.

Driver, R. D. (2012). *Ordinary and delay differential equations*. Springer Science & Business Media.

Ek, N. (1999). IEEE 802.1 p, QoS at the MAC level. *Apr*, 24, 0003–0006.

Ekelin, S., Nilsson, M., Hartikainen, E., Johnsson, A., Mangs, J. E., Melander, B. & Bjorkman, M. (2006, April). Real-time measurement of end-to-end available bandwidth using kalman filtering. *2006 ieee/ifip network operations and management symposium noms 2006*, pp. 73-84. doi: 10.1109/NOMS.2006.1687540.

Ericsson. (2016). Ericsson Mobility Report. Consulted at https://www.ericsson.com/mobility-report.

Ericsson AB, Huawei Technologies Co. Ltd, NEC Corporation, Alcatel Lucent, and Nokia Networks. (2015). Common Public Radio Interface (CPRI) specification V7.0. Consulted at http://www.cpri.info/downloads/CPRI_v_7_0_2015-10-09.pdf.

Ha, S., Rhee, I. & Xu, L. (2008). CUBIC: A new TCP-friendly high-speed TCP variant. *Sigops oper. syst. rev.*, 42(5), 64–74. doi: 10.1145/1400097.1400105.

Hanawa, T., Fujiwara, T. & Amano, H. (1996, Oct). Hot spot contention and message combining in the simple serial synchronized multistage interconnection network. *Parallel and distributed processing, 1996., eighth ieee symposium on*, pp. 298-305. doi: 10.1109/SPDP.1996.570347.

Holman, C., But, J., Branch, P. et al. (2012). The effect of round trip time on competing tcp flows.

Hubert, B. et al. (2002). Linux advanced routing & traffic control howto. *setembro de*.

IEEE 802.1Qau. (2010). IEEE Standard for Local and metropolitan area networks–Virtual Bridged Local Area Networks Amendment 13: Congestion Notification. *IEEE std 802.1qau-2010 (amendment to IEEE std 802.1q-2005)*, c1-119. doi: 10.1109/IEEESTD.2010.5454063.

IEEE Standard Association. (2011). IEEE standard for local and metropolitan area networks–media access control (mac) bridges and virtual bridged local area networks–amendment 17: Priority-based flow control. *IEEE std 802.1qbb-2011 (amendment to IEEE std 802.1q-2011 as amended by IEEE std 802.1qbe-2011 and IEEE std 802.1qbc-2011)*, 1-40. doi: 10.1109/IEEESTD.2011.6032693.

Institute of Electrical & Electronic Engineers. (2017a). IEEE 802.1CM Time-Sensitive Networking for Fronthaul. Consulted at http://www.ieee802.org/1/pages/802.1cm.html.

Institute of Electrical & Electronic Engineers. (2017b). P1914.1: Standard for Packet-based Fronthaul Transport Networks. Consulted at http://sites.ieee.org/sagroups-1914/p1914-1/.

Institute of Electrical & Electronic Engineers. (2017c). P1914.3: Standard for Radio Over Ethernet Encapsulations and Mappings. Consulted at http://sites.ieee.org/sagroups-1914/p1914-3/.

Institute of Electrical and Electronics Engineers. (2017). IEEE 802.1 Time-Sensitive Networking Task Group. Consulted at http://www.ieee802.org/1/pages/tsn.html.

Intel. (2014). DPDK: Data plane development kit. Consulted at http://dpdk.org/.

Jain, R. (1998). Myths about Congestion Management in High Speed Networks. *Corr*, cs.NI/9809088. Consulted at http://arxiv.org/abs/cs.NI/9809088.

Jeyakumar, V., Alizadeh, M., Geng, Y., Kim, C. & Mazières, D. (2014). Millions of Little Minions: Using Packets for Low Latency Network Programming and Visibility. *Corr*, abs/1405.7143. Consulted at http://arxiv.org/abs/1405.7143.

Jiang, J., Jain, R. & So-In, C. (2008, May). An explicit rate control framework for lossless ethernet operation. *2008 ieee international conference on communications*, pp. 5914-5918. doi: 10.1109/ICC.2008.1105.

Jiang, W., Ren, F. & Lin, C. (2015). Phase plane analysis of quantized congestion notification for data center ethernet. *Ieee/acm transactions on networking*, 23(1), 1-14. doi: 10.1109/TNET.2013.2292851.

Jose, L., Yan, L., Alizadeh, M., Varghese, G., McKeown, N. & Katti, S. (2015). High speed networks need proactive congestion control. *Proceedings of the 14th acm workshop on hot topics in networks*, (HotNets-XIV), 14:1–14:7. doi: 10.1145/2834050.2834096.

Kabbani, A., Alizadeh, M., Yasuda, M., Pan, R. & Prabhakar, B. (2010, Aug). AF-QCN: Approximate Fairness with Quantized Congestion Notification for Multi-tenanted Data Centers. *2010 18th ieee symposium on high performance interconnects*, pp. 58-65. doi: 10.1109/HOTI.2010.26.

Kale, P., Tumma, A., Kshirsagar, H., Ramrakhyani, P. & Vinode, T. (2011, June). Fibre channel over ethernet: A beginners perspective. *2011 international conference on recent trends in information technology (icrtit)*, pp. 438-443. doi: 10.1109/ICRTIT.2011.5972328.

Katevenis, M. (1997). Buffer requirements of credit-based flow control when a minimum draining rate is guaranteed. *The fourth ieee workshop on high-performance communication systems*, pp. 168-178. doi: 10.1109/HPCS.1997.864039.

Kelly, F., Raina, G. & Voice, T. (2008). Stability and fairness of explicit congestion control with small buffers. *Sigcomm comput. commun. rev.*, 38(3), 51–62. doi: 10.1145/1384609.1384615.

Lakshman, T. V. & Madhow, U. (1997a). The performance of tcp/ip for networks with high bandwidth-delay products and random loss. *Ieee/acm transactions on networking*, 5(3), 336-350. doi: 10.1109/90.611099.

Lakshman, T. V. & Madhow, U. (1997b). The performance of tcp/ip for networks with high bandwidth-delay products and random loss. *Ieee/acm transactions on networking*, 5(3), 336-350. doi: 10.1109/90.611099.

Lee, G. (2014). *Cloud networking: Understanding cloud-based data center networks*. Morgan Kaufmann.

Liu, J., Shroff, N. B., Xia, C. H. & Sherali, H. D. (2016). Joint congestion control and routing optimization: An efficient second-order distributed approach. *Ieee/acm transactions on networking*, 24(3), 1404-1420. doi: 10.1109/TNET.2015.2415734.

LTE, ETSI. (2009). Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception (3GPP TS 36.104 version 8.6. 0 Release 8). *Etsi ts*, 136(104), V8.

Marinos, I., Watson, R. N. M. & Handley, M. (2013). Network stack specialization for performance. *Proceedings of the twelfth acm workshop on hot topics in networks*, (HotNets-XII), 9:1–9:7. doi: 10.1145/2535771.2535779.

Mellanox. (june, 2014). Mellanox releases new automation software to reduce ethernet fabric installation time from hours to minutes. Consulted at http://ir.mellanox.com/releasedetail.cfm?ReleaseID=851785.

Mittal, R., Dukkipati, N., Blem, E., Wassel, H., Ghobadi, M., Vahdat, A., Wang, Y., Wetherall, D., Zats, D. et al. (2015). TIMELY: RTT-based congestion control for the datacenter. *Proceedings of the 2015 acm conference on special interest group on data communication*, pp. 537–550.

Mobile, C. (2013). C-RAN: the road towards green RAN. *White paper, ver*, 3.

Namba, S., Matsunaka, T., Warabino, T., Kaneko, S. & Kishi, Y. (2012, July). Colony-RAN architecture for future cellular network. *2012 future network mobile summit (futurenetw)*, pp. 1-8.

Network, I. (2010). Data Center Bridging (DCB) Congestion Notification (802.1qau). Consulted at http://blog.ipspace.net/2010/11/data-center-bridging-dcb-congestion.html.

Perry, J., Ousterhout, A., Balakrishnan, H., Shah, D. & Fugal, H. (2014). Fastpass: A centralized "zero-queue" datacenter network. *Sigcomm comput. commun. rev.*, 44(4), 307–318. doi: 10.1145/2740070.2626309.

Raina, G., Towsley, D. & Wischik, D. (2005). Part II: Control theory for buffer sizing. *Sigcomm comput. commun. rev.*, 35(3), 79–82. doi: 10.1145/1070873.1070885.

Satran, J. & Meth, K. (2004). Internet small computer systems interface (iscsi).

Shah, A. U., Bhatt, D. H., Agarwal, P. R. & Agarwal, P. R. (2012). Effect of packet-size over network performance. *International journal of electronics and computer science engineering*, 1, 762–766.

Snir, M. (2014). The future of supercomputing. *Proceedings of the 28th acm international conference on supercomputing*, (ICS '14), 261–262. doi: 10.1145/2597652.2616585.

So-In, C., Jain, R. & Jiang, J. (2008, June). Enhanced forward explicit congestion notification (e-fecn) scheme for datacenter ethernet networks. *2008 international symposium on performance evaluation of computer and telecommunication systems*, pp. 542-546.

Stephens, B., Cox, A. L., Singla, A., Carter, J., Dixon, C. & Felter, W. (2014, April). Practical DCB for improved data center networks. *Ieee infocom 2014 - ieee conference on computer communications*, pp. 1824-1832. doi: 10.1109/INFOCOM.2014.6848121.

Tanisawa, Y. & Yamamoto, M. (2013, Nov). QCN with delay-based congestion detection for limited queue fluctuation in data center networks. *2013 ieee 2nd international conference on cloud networking (cloudnet)*, pp. 42-49. doi: 10.1109/CloudNet.2013.6710556.

Tassiulas, L. & Ephremides, A. (1992). Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *Ieee transactions on automatic control*, 37(12), 1936-1948. doi: 10.1109/9.182479.

Tranter, W. H., Taylor, D. P., Ziemer, R. E., Maxemchuk, N. F. & Mark, J. W. (2007). Input Versus Output Queueing on a SpaceDivision Packet Switch. In *The Best of the Best:Fifty Years of Communications and Networking Research* (pp. 692-). Wiley-IEEE Press. doi: 10.1109/9780470546543.ch45.

Utkin, V. (1977). Variable structure systems with sliding modes. *Ieee transactions on automatic control*, 22(2), 212-222. doi: 10.1109/TAC.1977.1101446.

Valcarenghi, L., Kondepu, K. & Castoldi, P. (2017). Time-versus size-based cpri in ethernet encapsulation for next generation reconfigurable fronthaul. *Ieee/osa journal of optical communications and networking*, 9(9), D64-D73. doi: 10.1364/JOCN.9.000D64.

Varga, A. & Hornig, R. (2008). An overview of the omnet++ simulation environment. *Proceedings of the 1st international conference on simulation tools and techniques for communications, networks and systems & workshops*, pp. 60.

Varga, András and Hornig, Rudolf. (2008). An Overview of the OMNeT++ Simulation Environment. *Proceedings of the 1st international conference on simulation tools and techniques for communications, networks and systems & workshops*, (Simutools '08), 60:1–60:10.

Wan, T. & Ashwood-Smith, P. (2015, Dec). A Performance Study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv Enhancements. *2015 IEEE global communications conference (globecom)*, pp. 1-6. doi: 10.1109/GLOCOM.2015.7417599.

Wan, T., McCormick, B., Wang, Y. & Ashwood-Smith, P. (2016, Dec). ZeroJitter: An SDN Based Scheduling for CPRI over Ethernet. *2016 ieee global communications conference (globecom)*, pp. 1-7. doi: 10.1109/GLOCOM.2016.7842395.

Wan, T. & Ashwood, P. (2015). A performance study of CPRI over Ethernet. *IEEE 1904.3 task force*.

Wang, Z., Zeng, X., Liu, X., Xu, M., Wen, Y. & Chen, L. (2016). TCP congestion control algorithm for heterogeneous Internet. *Journal of network and computer applications*, 68(Supplement C), 56 - 64. doi: https://doi.org/10.1016/j.jnca.2016.03.018.

Wilson, C., Ballani, H., Karagiannis, T. & Rowtron, A. (2011). Better never than late: Meeting deadlines in datacenter networks. *Sigcomm comput. commun. rev.*, 41(4), 50–61. doi: 10.1145/2043164.2018443.

Wilson, S. (2008). *An adaptive packet size approach to tcp congestion control*. (Ph. D. thesis, University of British Columbia).

Zhang, Y. & Ansari, N. (2013). Fair Quantized Congestion Notification in Data Center Networks. *Ieee transactions on communications*, 61(11), 4690-4699. doi: 10.1109/TCOMM.2013.102313.120809.

Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M. H. & Zhang, M. (2015). Congestion control for large-scale RDMA deployments. *Sigcomm comput. commun. rev.*, 45(4), 523–536. doi: 10.1145/2829988.2787484.

Zhu, Y., Ghobadi, M., Misra, V. & Padhye, J. (2016). ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY. *Proceedings of the 12th international on conference on emerging networking experiments and technologies*, (CoNEXT '16), 313–327. doi: 10.1145/2999572.2999593.