Data Traffic Congestion Management for Data Center Ethernet

Fang Shuo

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of

Doctor of Philosophy

2013

Acknowledgements

First and foremost, I offer my sincerest gratitude to my supervisor, Dr. FOH Chuan Heng, who has guided and supported me throughout my Ph.D. candidature with his knowledge and patience, which also cultivates me from a novice to an independent researcher. I have not only acquired in-depth knowledge in my research area but also learnt to be a considerable and respectful person from him as my role model. I am also thankful for countless time and effort he spent on helping me improve my research skills. One simply could not wish for a better or friendlier supervisor.

Special thanks go to Dr. Khin Mi Mi AUNG, she introduced me to the world of storage area network. I am also indebted for her precious time for discussing with me over research issues and being supportive all the way here. I also greatly appreciate the chance to work with Dr. YU Yang and Ms. Renuga KANAGAVELU for their insightful opinions and discussions. Besides, my gratefulness goes to all other staff in Data Storage Institute, A*Star, Singapore, they have helped me in many ways in the preparation and completion of this study.

I would like to thank my family members, my grandmother and my parents, for supporting me throughout all my studies. Their dedication, love and persistent confidence in me, are the warmest support I ever received. I am very grateful to have them as my family.

Last but not the least, I owe my thanks to my boyfriend LI Lei, for his long time support with both his knowledge and love during my Ph.D. candidature, I am grateful to have him in my life.

Contents

List of Figures

List of Tables

Glossary

AIMD Addictive Increase and Multiplicative Decrease, page 2

ALR Adaptive Link Rate, page 37

ANSI American National Standards Institute, page 21

AQM Active Queue Management, page 2

BCN Backward Congestion Notification, page 23

CAPEX CAPital EXpenditure, page 1

CBR Constant BitRate, page 50

CC Congestion Control, page 66

CEE Converged Enhanced Ethernet, page 7

CP Congestion Point, page 26

CS Core Switch, page 50

DCB Data Center Bridging, page 7

DCBX Data Center Bridging Capabilities Exchange Protocol, page 28

DCE Data Center Ethernet, page 7

DCTCP Data Center TCP, page 25

DHT Distributed Hash Table, page 33

DLBMP Dynamic Load Balancing MultiPath, page 66

ECM Ethernet Congestion Management, page 21

ECMP Equal Cost Multipath, page 29

ECN Explicit Congestion Notification, page 24

EEE Energy Efficient Ethernet, page 36

ETS Enhanced Transmission Selection, page 21

F-ECN Forward Explicit Congestion Notification, page 25

FCoE Fibre Channel over Ethernet, page 6

FFD Fine Flow Differentiation, page 67

HBA Host Bus Adapters, page 5

HPC High Performance Computing, page 1

IB InfiniBand, page 5

IBoE InfiniBand over Ethernet, page 7

IEEE Institute of Electrical and Electronics Engineers, page 7

IETF Internet Engineering Task Force, page 15

INCITS InterNational Committee for Information Technology Standards, page 6

IPC Inter Process Communication, page 5

KVM Keyboards, Video and Mouse, page 5

L2MP Layer 2 MultiPath, page 8

LAN Local Area Network , page 1

LLDP Link Layer Discovery Protocol, page 28

LPI Low-Power-Idle, page 36

NIC Network Interface Card, page 5

OPEX OPerational EXpenditure, page 1

OSPF Open Shortest Path First, page 29

OSPF-OMP OSPF-Optimized MultiPath, page 31

PFC Priority-based Flow Control, page 21

QCN Quantized Congestion Notification, page 23

QoS Quality of Service, page 7

RIP Routing Information Protocol, page 29

RP Reaction Point, page 26

RTT Round Trip Time, page 24

SAN Storage Area Network , page 1

SDN Software-Defined Networking, page 35

SPB Shortest Path Bridging, page 29

SPOF Single Point Of Failure, page 9

STP Spanning Tree Protocol, page 1

TRILL Transparent Interconnection of Lots of Links, page 29

VLAN Virtual LAN, page 29

VLB Valiant Load Balancing, page 34

Abstract

Ten-gigabit Ethernet was first standardized in 2002. Over the years, the success standardization has led to wide acceptance and commercialization in the industry. Ethernet has offered an attractive solution for data center to consolidate heterogeneous traffic into a single fabric. A data center usually consists of three overlapped networks, Storage Area Network (SAN), Local Area Network (LAN) and High Performance Computing (HPC) Network. To satisfy their specialized requirements, each of them transmits traffic through dedicated fabrics. Replacing dedicated switches, transceivers and adapters with Ethernet components, the consolidation reduces both capital expenditure (CAPEX) and operational expenditure (OPEX). Although Ethernet exhibits many advantages, one must admit that it remains inadequate due to the lack of congestion management support. In this thesis, an Ethernet congestion management solution is proposed to satisfy requirements of loss-free, high throughput, and preferably, minimal energy usage for data center networks.

Conventional Ethernet only offers best effort delivery which tolerates frame drops, whereas SAN/HPC traffic is sensitive to loss. The consolidation solution should control congestion to ensure SAN/HPC traffic free from loss with higher priorities. Moreover, Ethernet uses Spanning Tree Protocol (STP) to prevent loops which prunes a mesh topology to a single tree structure eliminating redundancies and reducing network throughput. To preserve high bisectional bandwidth, the solution should enhance Ethernet with multipath feature to ease congestions. In addition, utilization of multipath, though increases network throughput, exerts extra power consumption by distributing traffic amongst all the paths when there is limited volume of traffic. The tradeoff of load balance or aggregation becomes

another research topic for adapting congestion with economizing data center cost. Thus a comprehensive Ethernet congestion management solution should satisfy three requirements, namely prioritize congestion control, high throughput congestion easing and energy efficient congestion adaptation. However, no systematic study has been found of such an approach.

This research is conducted to fill the gap of existing approaches and explore the possibility of an enhanced Ethernet congestion management for data centers. Firstly, a prioritized end-to-end congestion control scheme is proposed to combat congestion and protect traffic with higher throughput requirements, by introducing Active Queue Management (AQM) and Addictive Increase and Multiplicative Decrease (AIMD) to detect and constrain source rates towards congestions. Systematic analytical work is also presented to study suitability of the proposal and recommendation of system parameters. Secondly, an in-depth study is conducted to investigate on multipath routing, where a central controller overviews network status and controls generation and distribution of traffic. This two-tiered study not only effectively eases congestion by a fine grained load balance but also throttles excessive traffic from entering network core. Finally, being aware that load balance among multiple paths is at odds with efficient energy usage, the relationship between power consumption and number of active links is studied and found to be in a linear order. A proposal for energy optimization that dynamically balances load and adapts link rate with congestions is then presented to achieve minimal power consumption without compromising on throughput. All these three studies are validated by extensive simulations using OMNET++ simulator, which demonstrates the favorable performance of these schemes proposed. In summary, we propose prioritized congestion control that achieves ratio differentiation and combats congestion, loss-free multipath routing that distributes load almost equally to ease congestion and maximize network throughput, and efficient energy optimization that adapts load distribution and aggregation with congestion state. The congestion management solution proposed has achieved the objectives of the study with 3.5 SAN/LAN ratio, two-folded traffic volume support and up to 60% power savings.

Chapter 1

Introduction

Due to the growing number of data-intensive services in both enterprise computing and social computing, data centers are becoming more and more important in computer and Internet industry. They not only provide large enterprises with data storage and retrieval such as daily log backup but also support each individual to access and share information through web service providers. To satisfy the growing volume of data created and duplicated by various services, nowadays common data centers are facing performance concerns with massive number of servers and high requirement of bandwidth.

The major components of a data center are servers, storages and networking. Servers transform and process data, for example, by handling requests and operating web services. Storages such as disks, tape libraries, Redundant Array of Inexpensive Disks (RAID) [8] record information for later retrieval. Networking serves as communication medium moving data from one location to another. This thesis deals with networking components in data centers. Specifically, it explores the methodologies for improving data center performances from networking side.

The performance of most data centers is limited by their networking in terms of latency, throughput, not by servers or storages. In such a high-end system, most of the clock cycles are spent on wire delay, not gate delay [9]. As technology advances, memories and processors become small, fast and relatively inexpensive. The networking components, however, are scaling at a lower rate as compared to the complexity increase on other data center components.

At present, the largest data centers hold tens of thousands of servers, storages and switches, and the number is continuously increasing. Accordingly, deployment cost, power consumption and wiring complexity also increase. As a result, data centers not only spend tremendous expenditure for implementation and operation but also entail an enormous amount of human management. To mitigate these problems, many efforts have attempted to upgrade network performances, either in a greater step by consolidating hybrid network sectors, or in a detailed manner by using smarter algorithms and protocols in networking operations. In the macro level, data center network are replacing dedicated adapters and wirings for separate special-purpose network sectors to a single consolidate fabric, which is more economical in both maintenance and management. In the micro level, more concrete proposals for performance improvements on topologies, algorithms and protocols have also been developed.

In this chapter, firstly the concept of I/O consolidation is introduced to present its benefits for data centers. Secondly, the challenges and research issues of consolidation on Ethernet are discussed.

1.1 Data center I/O Consolidation

A data center preserves a heterogeneous nature that several networks overlap with each other. Servers and storages are required to install dedicated adapters and wirings to interconnect them with separate network fabrics. At the same time, these adapters and wirings take a significant proportion of operating cost for maintaining and cooling data centers. The complexity incurred by this heterogeneous nature not only results in management difficulties but also increases data center costs including both CAPEX and OPEX.

Considering these challenges, an ideal solution is to consolidate the heterogeneous network into one fabric. The desires for consolidation are not a newly raised topic. Rather, network designers and researchers have attempted to merge these networks into a single platform ever since the emergence of data centers. During these years, many potential fabrics have been examined and tested. Though they exhibit some good qualities, many issues are remained unsolved until the arrival of high speed Ethernet. With the evolvement of transmission rate to 10Gbps, 40Gbps and even 100Gbps, Ethernet can satisfy high performance communication requirement. Together with its low

cost and easy management, Ethernet is considered as a viable fabric for I/O consolidation.

1.1.1 Hybrid Data Centers

To satisfy a variety of purposes, a common data center system is usually composed of three network sectors that overlap and interconnect with each other, namely SAN, LAN and HPC. Most prominently and commonly, SAN links servers to storage devices as well as links storage devices such us tape libraries, RAID arrays one to another. Using Fibre Channel (FC) protocol sets for connectivity, SAN is characterized by high bandwidth and reliability with no frame loss.

LAN serves as management networks in data centers. The physical forms of management network are divided into two types. The first is a traditional short distance connection that links serial console ports of servers and keyboards, video and mouse (KVM) ports directly to KVM devices or indirectly through a management appliance. Thus, the management appliance provides KVM console access to the connected servers. A newer form of management connection running on Ethernet offers long distance KVM control over servers. Similar as a common IP network but running parallel to it within data centers, this Ethernet connection mostly works as a complement to the traditional connection by supplying out-of-band access to the servers for management as well as monitoring.

HPC overlays or partially replaces SAN that enables low-latency, high bandwidth data sharing among super computer systems. In data centers, HPC servers compute parallelly through Inter Process Communication (IPC). Interconnections for IPC fall into two approaches, some use systems such as Myrinet [10] to link nodes in a cluster. Others deploy InfiniBand (IB) for interconnections among processor nodes and high performance I/O nodes.

The separate designs of these overlapped network offer communication with specific purposes and transmission requirement. Nevertheless, this hybrid nature of data centers requires the systems to be installed with several types of adapters, and these adapters must be connected with their own type of wirings, to their own type of switches. To be specific, Network Interface Cards (NICs) are used for Ethernet connection, whereas Host Bus Adapters (HBAs) are required to transmit traffic via SAN. This separation on adapters, wirings and switches among heterogenous networks not only increases cost for

data center deployment, e.g. more adapters, server ports and more switches required, but also results in higher operational cost for maintaining and cooling data centers. Moreover, the complexity of interconnections also makes data center management a tough issue.

1.1.2 Step towards I/O Consolidation

Given the drawbacks of separating LAN, SAN and HPC, I/O consolidation has attracted attentions for years. The objective of I/O consolidation is to unify multiple types of traffic into a single transmission fabric. The industry has pursued several fabric convergence options, including IB and iSCSI. Each of these options has done well in its own segment in the IT market.

FC itself was proposed as one of the choices, but its limited support for mulitcast/broadcast traffic made it less credible. IB is a popular solution for I/O consolidation in the HPC field. However, it does not use FC protocols, the incompatibility with FC holds its penetration into a large market. Another attempt is iSCSI, its major drawback is that iSCSI runs SCSI command over TCP protocols. Using TCP as its underlayer protocol, iSCSI does not preserve the management and deployment model of FC. Therefore, it requires gateways, dedicated naming scheme and methods for zoning, which complicates the entire design.

As Ethernet transmission rate continually upgrades, many high performance applications no longer need to use a different technology for their interconnections. Although the specialized technologies still offer significant performance gains, Ethernet has exhibited many good features that satisfy most web application requirements. Statistics reveal that in 2005, 42% of the top 500 HPC sites use Gigabit Ethernet (GE) as their interconnection, nearly double the percentage that used it four years earlier.

With the commercialization of 10GE, and development of 40GE and 100GE in the near future, Fibre Channel over Ethernet (FCoE), aiming to use Ethernet technology to carry FC traffic [11], makes Ethernet a potential candidate as a unified fabric for consolidation. Developed by InterNational Committee for Information Technology Standards (INCITS) T11 group as part of the FC-BB-5, FCoE allows a single technology in the data link to operate in a data center, and makes a significant step towards I/O consolidation among LANs and SANs. It simplifies network topology while reducing cabling cost and complexity, eliminating half of the I/O adapter cards in a rack, reducing power

and cooling overhead. Alongside, proposals of InfiniBand over Ethernet (IBoE) [12] are also undergoing. Enabling FCoE and IBoE technologies, Ethernet is able to carry FC and IB traffic as a consolidating fabric.

This consolidation offers many good qualities including reduction, simplification and standardization of cabling, absence of bottleneck gateways, less power for maintenance and cooling, as well as reduced deployment cost. Taking a typical VMware server as an example, it normally requires $6\times$1GE ports, $2\times$4GFC ports and 8 cables. Transforming to a FCoE fabric, the number has reduced to $2 \times$ 10GE ports which increases the total bandwidth available as well as greatly reduces infrastructure. Besides it also reduces the 4 top-of-rack switches to 2 FCoE switches.

Not only research and academic sides promote the idea of employing Ethernet as a unified fabric, industries also put their efforts towards FCoE commercialization. Many key players in the field have launched their new FCoE enabled products, including Cisco Nexus 5000 and 7000 series, Brocade 8000, HP 2408 and etc. With the support from industrial field, data center steps further into a realistic I/O consolidation.

1.2 Technical Challenges and Opportunities of Ethernet Consolidation

Admittedly, Ethernet exhibits many good features as a unified fabric, it still remains incomplete with a series of Quality of Service (QoS) requirements to be addressed, among which congestion management is one of the top issues. Given data center characteristics, Ethernet congestion management should consider priority differentiation, high bandwidth routing and efficient power consumption. Institute of Electrical and Electronics Engineers (IEEE) has formed a task group, Data Center Bridging (DCB) [13], targeting these Ethernet Enhancements. Similar scopes have been investigated by IBM, termed as Converged Enhanced Ethernet (CEE) [14], and Cisco, termed as Data Center Ethernet (DCE) [15]. The three terms are interchangeably used to refer the same architecture in this thesis, however, in most cases, the author prefers to use the term DCE for its clear differentiation with conventional Ethernet.

A well-designed converged fabric should keep hybrid traffic running smoothly without interferences among each others. Ethernet, as an economical choice with compatibility over other network sectors, is originally designed to transmit best effort traffic

dedicated for LAN. The major challenge of I/O consolidation is to satisfy the requirements of different traffic classes with a single network. The newly developed Ethernet protocol demands enhancements on congestion management with prioritized traffic, layer 2 MultiPath (L2MP) and energy efficient concerns to ensure lossless traffic with low latency and high throughput.

Fundamentally, FC, Ethernet and IB are developed to handle different characteristics of traffic and have different considerations for traffic transportation. Precisely, FC and IB technologies are designed to achieve high speed, low latency and lossless packet transportation that well suits computing clusters and SANs, whereas Ethernet is designed to carry best effort traffic. In the current design, FC and IB use link level flow control that results in strict admission control to prevent traffic congestion from happening [16], while Ethernet provides connectionless service with minimum control on the traffic flow and no control on the traffic congestion. Ethernet that puts FC traffic which requires lossless transportation onto Ethernet which only offers connectionless service will result in serious performance degradation due to the inadequate handling of FC and IPC traffic by Ethernet.

Apart from lossless requirement, three research issues need to be addressed. Firstly, Ethernet should differentiate congestion control among various priorities. With the introduction of a consolidated fabric, a mix of data center traffic such as SAN, LAN and IPC will be running simultaneously. Owing to the different characteristics and importance of various classes of traffic, differentiated handling of packet transportation should be considered. This gives rise to the need for traffic prioritization and service differentiation. Currently, one available mechanism is due to IEEE 802.1Q [17], dealing with the traffic prioritization and service differentiation by defining a tag field to differentiated priorities which maps IP traffic to Priority one, FC traffic to Priority three, and IPC traffic to Priority Six. Yet its priority control remains at a coarse grained level without precise flow control.

Secondly, Ethernet should offer high throughput by increasing bisectional bandwidth. Conventional Ethernet utilizes spanning tree protocol (STP) [18], [19] for frame forwarding, which always reduces a meshed network topology to a tree by blocking some links and ports from forwarding. STP guarantees that only one path can exist between any two communicating points and thus forwarding loops are eliminated from the network. However, STP eliminates the multipathing capability and shows

poor bandwidth utilization for such a meshed Ethernet. On the contrary, data center, inherently designed with redundancies for fault tolerance, would receive a significant reduction on bisectional bandwidth if deployed with STP. The pruned tree structure and poor bandwidth utilization likely leads to congestion, especially at the core as those switches need to carry aggregated traffic from all end devices in the network. By fully utilizing network redundancies, an appropriate proposed L2MP protocol can ease congestions while providing high network throughput.

Thirdly, Ethernet should explore the dynamic range of energy usage to provide a more energy-efficient network. The introduction of the concept of I/O consolidation is partially due to the purpose for power reduction, with less switches and less cabling. Although reducing power consumption incurred by excessive cabling, network still has the potential of further saving from two aspects. Firstly, more savings can be incurred by adaptively utilizing redundancy. To ensure data availability, a large part of data center infrastructure serves as redundancies to prevent single point of failure (SPOF). Optimizing the redundant portion of power consumption without compromising network performances can probably be achieved by dynamical distribution of traffic. Secondly, more savings can be exploited by adaptively switching Ethernet link rates. As a mature technology with years development, Ethernet has evolved from a very low transmission rate to even 40Gbps/100Gbps. Many switch models are also compatible with lower data rates as 10Mbps and 100Mbps which consume less power. Potentially, the various levels of power consumption indicate a flexibility of energy usage within DCE. The application of Ethernet as a unified fabric not only meets the original aim but also offers an opportunity for further minimizing power consumption, yet literature that explores this area of research remains inadequate.

1.3 Thesis Organization

To address the above issues discussed in Section 1.2, the principal focus of my research is to provide a data center congestion management solution with delivery differentiation, high throughput and low energy consumption. According to data center components, a comprehensive congestion management solution should consider three major parts, namely sources, switches and physical links. **Sources**, in data centers, are technically named as zone members including initiators, for example servers that transform

and process data by handling requests and operating web services, and targets, commonly data storages such as disks, tape libraries, RAID [8] that record information for later retrieval. **Switches** are in the middle of data sources, they function to switch packets and make routing decisions that transmit traffic among communication parties. **Links** serve as communication medium transmitting data from one location to another, they connect sources and switches, or among switches. Managing these three network components can comprehensively control the network performances, thus achieve the objectives of this study.

The main body of the thesis is divided as follows.

Chapter 2 presents an in-depth literature review of technologies in data center network. It summarizes mainstream standards regarding this research scope, proposed data center topologies as well as source, switch and link control technologies and algorithms.

Chapter 3 proposes a new source control scheme which demonstrates a differentiated congestion control for storage, IP and HPC traffic running on Ethernet. Employing control theoretic as an analytical tool, this chapter also offers detailed analytical results and guidelines for satisfying priority requirements. The scheme proposed in this chapter is also integrated with multipath capability as discussed in Chapter 4

Chapter 4 describes a dynamic switch control scheme which is designed for easing congestions by balancing load distribution among multiple paths. In this scheme, we employ OpenFlow technology by decoupling control information from conventional data path to a distinct central controller, which significantly reduces control overhead.

Chapter 5 provides our observation on data center traffic characteristics and discusses link control for reducing energy usage. Our scheme for optimizing power consumption within data centers is proposed by adapting active number of links and link rates according to congestion status.

Finally, Chapter 6 concludes the thesis with contributions and future work. It summarizes the research findings and our contribution achieved so far on this data center congestion management topic, and further discusses the future work.

1.4 Contribution and Publication List

The contribution of this thesis is threefold. Firstly, we introduce a new source control scheme for differentiated congestion control. The scheme utilizes asymmetric AIMD rate controls at the sources to achieve differentiated bandwidth usage and congestion handling. By using control theoretic analysis and computer simulation, we confirm the effectiveness of our proposed scheme.

Secondly, we propose a switch control scheme that offers multipath capability for data centers. The scheme assigns traffic flows to lightly loaded paths to ease congestions. We further extend it to include source congestion control to result in a new joint source-switch congestion control mechanism. Under light to moderate load situation, the source-switch congestion control scheme detects and avoids congestion. Under heavy to overloading situation, our scheme regulates the sources to prevent excessive traffic from entering the network. We demonstrate using computer simulation that it is practical and effective to increase network throughput.

Thirdly, we investigate data center traffic characteristics and switch power consumption level. Our investigation has led to a proposal of a link control scheme for reduced energy usage. By dynamically adjusting routing, data center network can consume energy proportionally to traffic volume. Using greedy approach, we develop a traffic aggregation algorithm to reduce number of operating switches and links under light load. We show that a more energy efficient data center can be achieved through our scheme.

This thesis is based on research papers, which are listed below in chronological order.

Journal Papers

- S. Fang, C. Foh, and K. Aung, "Differentiated congestion management of data traffic for data center Ethernet," *IEEE Transactions on Network and Service Management*, no. 99, pp. 1–12. Related content of this paper is presented in Chapter 3.

- Y. Yu, S. Fang, K. Aung, C. Foh, H. Li, and Y. Zhu, "A Layer-2 Multipath Solution and its Performance Evaluation in Data Center Ethernets," to appear *International Journal of Communication Systems*.

- S. Fang, Y. Yu, C. Foh, and K. Aung, "A Loss-free Multipathing Solution for Data Center Network Using Software-defined Networking Approach," to appear in *IEEE Transactions on Magnetics*. Related content of this paper is presented in Chapter 4.

Conference Papers

- S. Fang, C. Foh, and K. Aung, "Differentiated Ethernet Congestion Management for Prioritized Traffic," in *IEEE International Conference on Communications (ICC), 2010*, pp. 1–5. Related content of this paper is presented in Chapter 3.

- S. Fang, C. Foh, and K. Aung, "Prompt Congestion Reaction Scheme for Data Center Network using Multiple Congestion Points," in *IEEE International Conference on Communications (ICC), 2012*, pp. 2712–2716.

- S. Fang, H. Li, C. Foh, Y. Wen and K. Aung, "Energy Optimizations for Data Center Network: Formulation and its Solution," to appear in *IEEE Global Communication Conferences (Globecom), 2012*, pp. 3280–3285. Related content of this paper is presented in Chapter 5.

- S. Fang, Y. Yu, C. Foh, and K. Aung, "A Loss-free Multipathing Solution for Data Center Network Using Software-defined Networking Approach," to appear in *Storage System, Hard Disk and Solid State Technologies Summit, IEEE APMRC, 2012*. Related content of this paper is presented in Chapter 4.

Chapter 2

Background and Literature Review

As we have discussed in Chapter 1, a comprehensive and efficient data center congestion management solution consists of three schemes, namely source control, switch control and link control, which correspond to the three major components of a data center as shown in Fig. 2.1. Source control serves as admission control which avoids excessive traffic from entering the network. It can differentiate priorities by throttling excessive traffic with different rates in a fine grained manner. However, source control does not fit for routing, since sources such as end stations lack whole network information depending on which routing decisions are made. On the other hand, switches connect each other and form network mainframe, such that they can gather network status easily for routing. As a result, switch control is suitable for congestion easing by implementing multipath routing and distributing traffic properly. Although overseeing the network as a whole picture, switches can hardly provide flow granularity controls on priorities as sources do, or adapt to congestions with a part of the network, such as a single link. Link control focuses on a particular part in a network, which adaptively changes the number of active links and link rates, and improves energy efficiency according to congestion levels. In the following, a thorough literature review will be presented from these three control schemes.

In this chapter, we first present a standard data center topology and some emerging topologies which are the basis of the following discussions. Some of newly proposals depend on the emerging topologies that cater to one or more data center requirements.

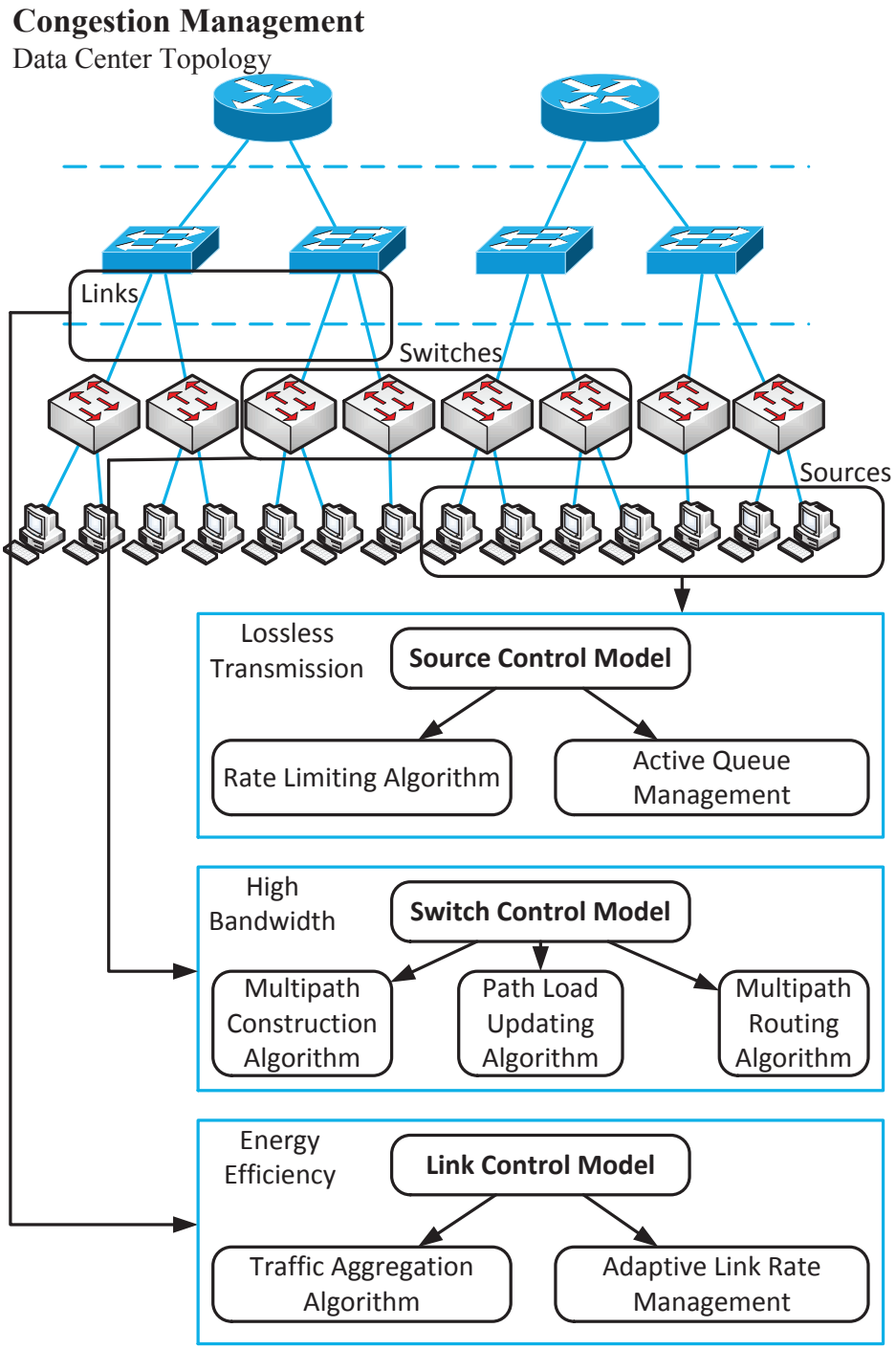


Figure 2.1: Overview of a comprehensive congestion management solution for data center.

In the following sections, we review existing literature from three perspectives of source, switch and link control schemes. In each section, our discussion follows the outline that firstly introduces standards of data center protocols, then describes related research techniques and lastly presents recent data center proposals. First, Ethernet Enhancement standards have been developed under several international task groups, including Internet Engineering Task Force (IETF) and IEEE. Particularly, IEEE 802.1 Data Center Bridging (DCB) task group [13] deals with the main Ethernet Extensions for data centers. Through standardizing associated network protocols, DCB can upgrade Ethernet for better data center performances. Second, mature techniques in other research areas of network, such as TCP or Internet technologies, are demonstrated. With similar scopes as each of the three control schemes, these techniques can be adapted in data centers as potential approaches. Last, researchers from either industrial field or academic field also devote their time and intelligence proposing innovative ideas to improve data center performance. The newly raised proposals are another topic in each section.

2.1 Data Center Topologies

Network topology refers to the static arrangement of links and nodes in an interconnection network, the road over which packets travel. Selecting a network topology is the first step in designing a network. Both congestion easing and congestion adaptation depend heavily on network topology.

In this section, the standard approach is firstly introduced, followed by demonstration of other emerging approaches. In data centers, a typical topology follows a tiered tree structure which aggregates bandwidth from bottom up. To handle the aggregated traffic, high end switches are required in upper layers. As a data center scales up, higher end switches with more capacity requirements are deployed. That means, the scalability of a data center is constrained by switch capacities. Once there is no higher end switches, the limit of a data center is reached. To mitigate this problem, some emerging approaches focus on transforming conventional data center topology into other new types. Most of them not only try to solve the scalability issue by using commodity off-the-shelf switches, but also satisfy one or more other data center requirements as we discuss in next sections.

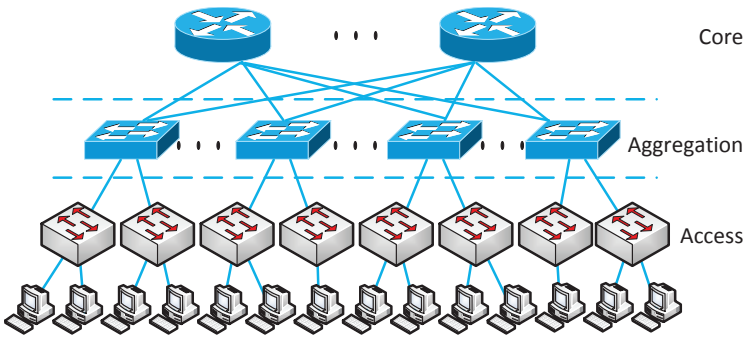


Figure 2.2: Typical data center interconnection topology.

2.1.1 Standard Approach

The multi-tier model is the most common topology used in data centers [20], [21]. A typical three-tier communication infrastructure of a data center is shown in Fig. 2.2, which is comprised of access layer connected to servers, aggregation layer for aggregation of access switches and core layer serving as the gateway to where other modules connect, including, for example, the extranet, WAN, and Internet edge. Under this infrastructure, end hosts connect to 1G Top of Rack (ToR) switch, i.e. access switch, and ToR switches connect to 10G End of Row (EoR) switches, i.e. aggregation switch. For large clusters, aggregation switches connect to 10G core switches. The oversubscription ratio of this infrastructure is typically from 2.5:1 to 8:1. As data centers scale to host more servers and switches with higher transmission rate, this classical model faces many challenges in terms of performance, cost, routing etc.

2.1.2 Emerging Approaches

Telecommunication networks adopt butterfly topology to provide the shortest switching path, and Clos network to satisfy non-blocking communication. Derived from these two classic topologies, the newly proposed flattened butterfly [2] and Fat Tree topologies inherit their advantages and adapt them with data center requirements. Another innovative research trend is to recursively construct network from a basic unit [4], [5], which mainly addresses network scalability issue. In the following we discuss these three proposals respectively.

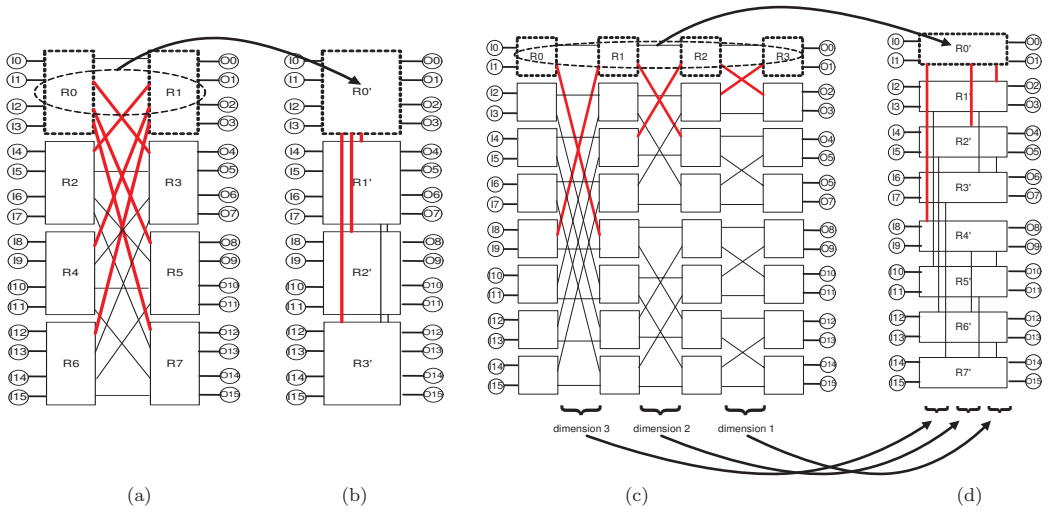


Figure 2.3: Block diagram of butterfly and flattened butterfly networks (Fig. 1 of [2]).

2.1.2.1 Butterfly Networks

A k-ary n-fly Butterfly network consists of k^n source terminal nodes, n stages of k^{n-1} $k \times k$ crossbar switch nodes, and finally k^n destination terminal nodes. Examples of butterfly networks are shown in Fig. 2.3. In an N-node butterfly network, any node can be reached within $H = \log_k N + 1$ hop counts, which is called network diameter. The major feature of butterfly topology has the minimum diameter for a N-node network.

Although optimal diameter is an attractive feature, butterfly networks obtain these feature at the expense of losing path diversity and locality, which gives rise to a newly flattened butterfly proposal. It transforms from a conventional butterfly network by 'flattening' each row, replacing switches on the row with a single crossbar switch. A flattened butterfly network is denoted as by (k, n), where parameters of k and n are inherited from conventional butterfly, but in the derived flattened butterfly the definitions become confusing. We can simply perceive it as that in a k-ary n-flat flattened butterfly network consists of $n - 1$ dimensions each with k switches.

As shown in Fig. 2.3 [2], derivations of butterfly and flattened butterfly networks are illustrated. Fig. 2.3 (a) and (b) specify a 4-ary 2-fly butterfly and a corresponding 4-ary 2-flat flattened butterfly with a single dimension, and Fig. 2.3 (c) and (d) specify a 2-ary 4-fly butterfly and a corresponding 2-ary 4-flat flattened butterfly. Lines denote unidirectional links in the butterfly and bidirectional links (i.e. two unidirectional links)

in the flattened butterfly.

While previous works on energy efficient data center only focused on reducing energy of links or switches, Abts *et al.* [22] view the picture from network level. They point out a data center network based on the flattened butterfly topology results in a more power efficient network [2], [22].

Flattened butterfly topology may present as an energy efficient proposal, but it requires high end switches with more switch ports as compared to common butterfly or other topologies. The problem can be addressed by adding extra dimensions to the butterfly, but compromising path length at the expense.

2.1.2.2 Clos Networks

A Clos network is a multi-stage network in which each stage is composed of a number of crossbar switches. Typically, three stages are used and Clos networks with any odd number switches can be recursively derived from the three-stage Clos by replacing the middle stage with three-stage Clos networks.

A symmetric Clos is characterized by a triple, (m, n, r) where m is the number of middle-stage switches, n is the number of input/output ports on each input/output switch, and r is the number of input/output switches. Asymmetric Clos networks are rarely used, in which the r and n differ between the input and output stages.

A particular folded-Clos network, Fat Tree, attracts attention these years for increasing bisectional bandwidth without oversubscriptions. Al-Fares *et al.* [3] advocate on Fat Tree topology to provide high level of bandwidth for many end devices by appropriately interconnecting commodity switches, shown in Fig. 2.4. A more detailed description of Fat Tree will be provided in the Chapter 5 in Section 5.2.1. Later, instead of using conventional flat addresses, PortLand [23] updates this design by creating an addressing scheme which follows a hierarchical fashion and specifies pod ID, switch ID and host ID in separate fields of an address. PortLand also proposes to use two-level routing tables and spread outgoing traffic on the low-order bits of the destination IP addresses, and such mechanisms can significantly reduce the number of entries in a routing table. However, the design of PortLand simply employs a static routing algorithm that routes traffic regardless of the volume on each route. Particularly, it uses hash function to map traffic to routes only based on the destination addresses and neglects the real-time traffic load situation. Without dynamic load monitoring and balancing,

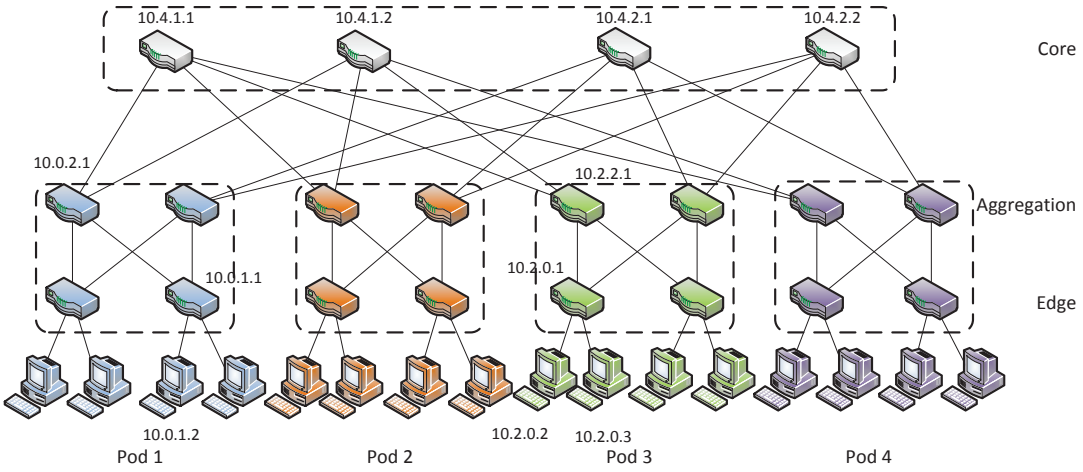


Figure 2.4: Illustration of Fat Tree topology (Fig. 3 of [3]).

the significance of PortLand in achieving fair link utilization and network scalability is in doubt.

In this thesis, we shall focus our research mainly on tree topology, or Fat Tree topology for network redundancies, which are closer to a typical data center topology with minimal change on the existing infrastructure than the other emerging topologies discussed in this section (Section 2.1).

2.1.2.3 Recursive Topologies

Guo *et al.* propose a novel DCell [4] structure for Data Center Network, in which a higher level DCell is constructed from lower level DCells. The basic building block $DCell_0$ consists of n equally-acted servers and a mini-switch as shown in Fig. 2.5. This proposal exhibits desired characteristics such as double exponential scalability, fault-tolerance and higher network capacity. Later, they also propose another recursive topology called BCube [5], which is specially designed for shipping container based, modular data centers. In this design, a server performs not only as an end host but also as an intermediate relay node. Likewise, $BCube_0$ also consists of n servers and a n-port switch. More generally, a higher level $BCube_k$ consists of n $BCube_{k-1}$s and n^k n-port switches, as shown in Fig. 2.6. This structure provides good support for speeding up one-to-x/servaral/all traffic pattern which is required by many data center applications, and maintains high capacity for all-to-all traffic. However, both DCell

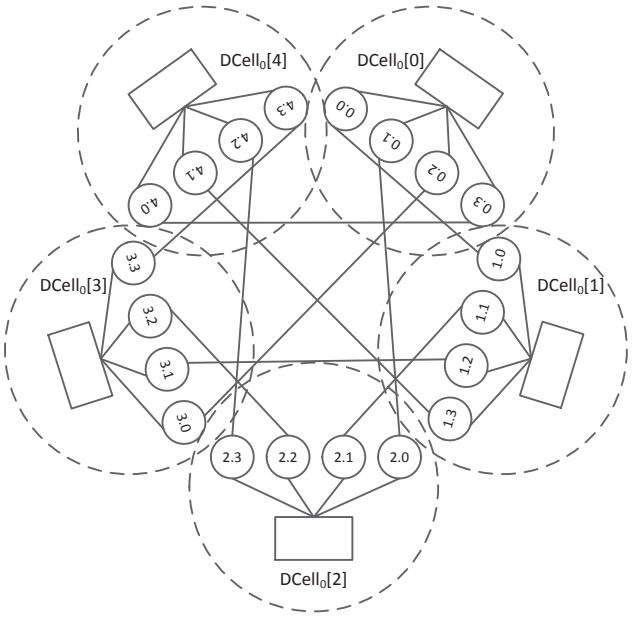


Figure 2.5: Illustration of DCell topology (Fig. 1 of [4]).

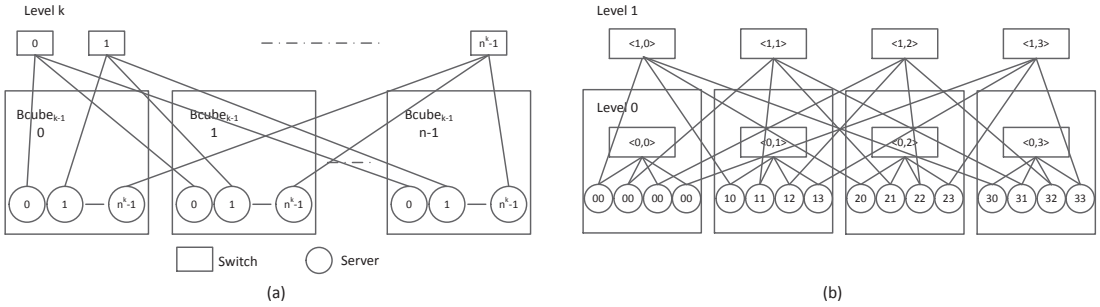


Figure 2.6: Illustration of BCube topology (Fig. 1 of [5]).

and BCube involve multi-NIC servers and can lead to high link oversubscription.

2.2 Data Center Source Control

While a topology determines the ideal performance of a network, flow control is one of the two key factors that determine how much of this potential is realized [24]. We shall discuss the other key factor, routing, in Section 2.3.

The flow control objectives for data center Ethernet include supporting lossless transmission for data traffic as well as differentiating different traffic classes with pri-

orities [21].

There are two main approaches for preventing frame drop. One approach adopts buffer-to-buffer flow control, in which the control intelligence is within switches. The other approach uses end-to-end congestion control, which equips the control unit at sources. End-to-end control prevents congestion more efficiently by holding the excessive traffic from entering network core and avoids spread of congestion as buffer-to-buffer flow control does.

Additionally, prioritized flow control is a new requirement for a consolidated data center network. Several standards are announced to support differentiated service including Priority-based Flow Control (PFC) and Enhanced Transmission Selection (ETS).

In the following, we firstly describe related standards for flow control. Control techniques such as AIMD and AQM are then discussed, which serve as basis for a variety of congestion control proposals including our scheme presented in Chapter 3. Finally, discussions and evaluations on two source congestion control proposals are presented. Especially, Ethernet Congestion Management (ECM) is highlighted as one of the important solutions and the prototype of our proposal in Chapter 3.

2.2.1 Standardizations

Given the objective of source control to control congestions, in this section, we shall present common congestion control standards for SAN and LAN, as well as newly developed flow control standards for data centers respectively.

FC was standardized in the T11 Technical Committee of INCITS, an American National Standards Institute (ANSI)-accredited standards committee. Traditionally, SANs based on FC technology use buffer-to-buffer **credit-based flow control** to avoid traffic congestion in the network. On a data link, the amount of traffic that a sender can transmit depends on the amount of credits allocated by the receiver. The receiver can prevent excessive incoming traffic by a proper credit allocation scheme, and hence traffic congestion can be avoided.

On the other hand, conventional Ethernet achieves the lossless feature by **PAUSE mechanism** [25]. The introduction of DCE has triggered demands for Ethernet to deal with traffic congestion. An immediate solution to enable equivalent credit-based flow control on the Ethernet is to employ PAUSE mechanisms [26].

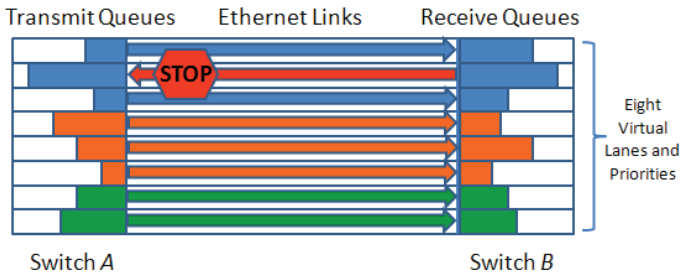


Figure 2.7: Illustration of IEEE 802.1Qbb [6] Priority Flow Control.

It is suggested that a proper implementation of PAUSE mechanism may enable lossless transportation within an Ethernet network. In PAUSE mechanism, a downstream port may issue a PAUSE frame to halt data transmission from the upstream port for a specified period of time to avoid traffic congestion. However, stopping transmission of an upstream port might lead to further congestion points in its predecessors, and continually, traffic congestions can spread to all upstream ports even for those that do not transmit excessively. In other words, rather than dealing with the source of the problem, this mechanism penalizes all sources when a traffic congestion event occurs. Additionally, due to the buffer-to-buffer control, the control of a congestion and the recovery from a congestion are done hop-by-hop which takes some time to converge. Moreover, the existing PAUSE mechanism does not offer differentiated control for different traffic types.

PFC [6] has extended PAUSE mechanism on per-priority basis, which enables a fine-grained flow control with coexistence of eight different classes of traffic as defined in IEEE 802.1p [27] for priority based forwarding. The IEEE 802.1Qbb [6] standard offers a link level flow control mechanism that can be controlled independently for each frame priority by defining a tag containing three-bit priority field. According to the priority field, PFC maps traffic classes to different priorities and prevents traffic interference. However, apart from offering differentiated congestion control, PFC inherits all other shortcomings of PAUSE mechanism. Illustration of mechanism of PFC is shown in Fig. 2.7.

While the aforementioned standards are based on buffer-to-buffer flow control, the following two standards deal with different perspectives, either a coarser grained control of bandwidth, or a finer grained of end-to-end flow adjustment.

ETS specifies a common management framework for assignment of bandwidth to frame priorities in IEEE 802.1Qaz [7]. With the structure shown in Fig. 2.8, it is feasible to assign bandwidth to each priority group, e.g. 40% of LAN, 40% of SAN and 20% of HPC. Although this scheme could control the bandwidth sharing on a coarse grain level, multiple classes and application flows are sharing bandwidth of the group with the same priority. Hence, flow level congestion control is necessary as a supplement to ETS.

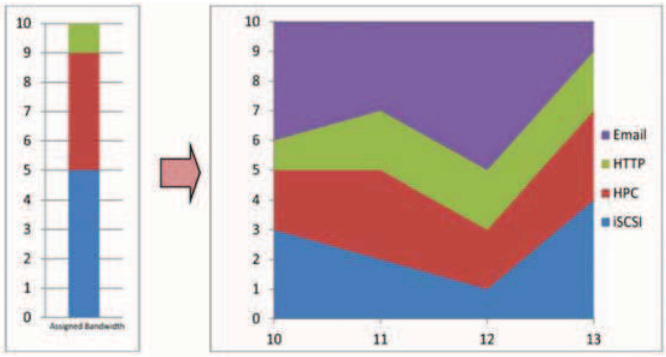


Figure 2.8: Illustration of IEEE 802.1Qaz [7] Enhanced Transmission Selection.

To avoid congestion from spreading across the entire network, Cisco has proposed **ECM** as a Layer-2 end-to-end congestion notification protocol. Previously called Backward Congestion Notification (BCN) and later known as Quantized Congestion Notification (QCN) with significant improvements over BCN, it has been standardized in IEEE 802.1Qau [28]. Implementing rate limiters at sources, ECM can hold excessive traffic at the edge of a network. With less number of traffic flows, the flows that caused congestion can be easily constrained and regulated. However, the key feature of differentiated control remains missing from ECM. Since our prototype is closely correlated with ECM, a more detailed description of ECM will be provided in the Section 2.2.3.2.

2.2.2 Congestion Control Techniques

A well-designed source control scheme in data centers aims to hold excessive traffic from entering the network and thus prevents frame drop. To achieve this, end-to-end approach in flow control is more suitable as compared to buffer-to-buffer approach. A common end-to-end approach consists of two parts, a rate adjustment algorithm at

sources and a congestion detection mechanism at switches/routers. Matured technologies of AIMD is widely used for tuning transmission rate and AQM is sufficient for congestion detection.

On the source side, **AIMD** is implemented for rate adjustment. As a widely used algorithm in congestion management, it is best known in TCP congestion avoidance. Depending on whether using a window-based AIMD or rate-based AIMD, either TCP congestion window size or transmission rate performs an increase-by-one and decrease-to-half algorithm. To be specific, a source keeps increasing its window size (or transmission rate) linearly until loss occurs indicted by TCP acknowledgement after a round trip time (RTT). Once loss is detected, window size (or transmission rate) is cut to half. Later versions of TCP consider the increase value and decrease ratio as parameters for the congestion window adjustment strategy, hence generalizes the algorithm as AIMD.

AIMD and its throughput performance have been a fertile area of research in the literature. An important study on AIMD for TCP congestion control is due to Padhye *et al.* [29], where a formulation to derive the mean window size given the packet loss rate under a certain AIMD setting is provided. Based on this result, bandwidth sharing behavior among several TCP connections of a symmetric setting [30], [31], [32] as well as an asymmetric setting [33], [29] are given. It has also been shown that with certain AIMD settings on several connections, different throughput levels can be achieved [34]. Therefore, AIMD not only serves as a congestion control mechanism but also satisfies traffic priority differentiation on throughput in DCE.

On the switch/router side, **AQM** technique takes preemptive actions prior to a potential buffer overflow in a queue. Conventionally, a router or other networking device buffers to its maximum capacity and simply drops packets when the queue is full. Nevertheless this can lead to a global synchronization of reducing and progressing simultaneously, thus network becomes under-utilized and flooded by turns. AQM algorithms such as *Random Early Detection* address this issue by monitoring queue length and dropping packets based on statistical probabilities.

AQM algorithms are mainly divided into early dropping and *Explicit Congestion Notification* (ECN) marking on packets before a switch's buffer or a router's queue is full. Dropping packets is widely used as an indicator of network congestions before ECN is introduced. Without dropping, an ECN-aware switch/router marks a bit in the IP header to signal potential congestion. The receiver of the packet echoes the

congestion indication to the sender, which reduces its transmission rate as though it detects packet drop.

2.2.3 Existing Proposals

As DCE requires strict congestion control schemes with no traffic loss, ECN that uses marking of AQM to notify network congestion without dropping packets is appropriate for lossless packet transportation. Important studies on data center network congestion management based on ECN include BCN [35], later enhanced versions of ECM [36], QCN [37], details of this work will be discussed next in Section 2.2.3.2. Based on BCN, Jiang *et al.* propose Forward Explicit Congestion Notification (F-ECN) [38], subsequent enhanced version of E-FECN [39], and *Data Center TCP* (DCTCP) [40] which is a newly proposed work to address drop prevention in shallow buffer switches on TCP layer.

2.2.3.1 Data Center TCP (DCTCP)

Recently, a cross-layer congestion control, named data center TCP or DCTCP in short, operating at Layer-4 for data center networks has been proposed [40]. DCTCP proposes marking of ECN based on Layer-2 buffer state and a new rate adjustment scheme replacing AIMD. While DCTCP operates at Layer-4, with the introduction of ECN based on Layer-2 buffer state, it is capable of reacting to a potential congestion before a congestion event occurs. However, no differentiated congestion control is introduced in DCTCP. Moreover, the relatively long host-to-host turnaround time at Layer-4 has created a lag in the control.

2.2.3.2 Ethernet Congestion Management (ECM)

FC technology is currently a solution for SANs. FC technology offers lossless transportation between hosts and storages, and this lossless transportation helps SAN achieve high performance. Enhancing existing FC technology with FCoE immediately causes problem as Ethernet technology is currently incapable to deliver traffic with lossless demand. Designing a congestion control for an FCoE based SAN must ensure that Ethernet takes preemptive actions to prevent congestion and buffer overflow from occurring. This leads to the proposals of implementing a particular ECN at switches

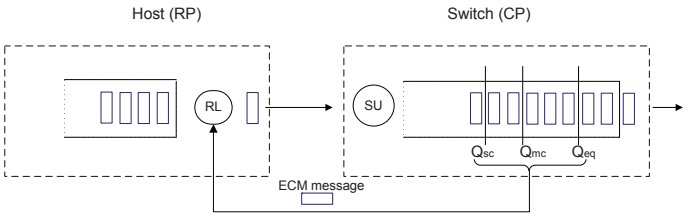


Figure 2.9: System model of ECM.

and rate limiters at hosts. This design allows excessive traffic to be held at the hosts, which pushes congestions from the core to the edges of the network. Precisely, when the current buffer utilization reaches a certain level, the switch acting as the congestion point (CP) notifies the host acting as the reaction point (RP) about the status, and the rate limiter in the host reduces its sending rate accordingly as shown in Fig. 2.9 [41]. In the following, we describe the current ECM proposal.

In ECM, a switch consists of a CP that samples incoming frames at a certain probability. Whenever a frame is sampled, the buffer utilization is also checked. This checking is designed to ensure that relevant ECN actions can be taken when traffic arrivals threaten buffer overflow. The probability P of sampling an incoming frame is critical to ensure fast reaction to congestion without excessive computational overheads. In [41], the measure of sampling probability is described by sampling rate where a sampling is executed on every $E[L]/P$ received bytes, here average frame length is defined as $E[L]$.

The CP checks the buffer utilization against three thresholds, namely, Q_{eq}, Q_{mc} and Q_{sc}, where $Q_{eq} < Q_{mc} < Q_{sc}$. Based on the buffer utilization, corresponding ECM message is sent from the CP to the RP whose arriving frame is last sampled at the CP. Based on the received ECM message, the rate limiter adjusts the transmission rate of a source accordingly to let the buffer operate around the desired equilibrium level. Table 2.1 shows the relationship among thresholds, ECM messages and the reactions of a rate limiter.

Based on the three thresholds described in Table 2.1, a CP is said to operate in three states, namely equilibrium, mild congestion and severe congestion. When the buffer of a CP is utilized below Q_{mc}, the CP is said to operate in the equilibrium state. In this state, an ECM(Q_{off}, Q_{delta}) message is used for notification. This message contains two parameters, where Q_{off} is the offset of the current buffer utilization with respect to Q_{eq},

Table 2.1: Functionality of the thresholds and corresponding messages

Label	Threshold Name	Message	Reaction
Q_{eq}	Equilibrium	ECM(Q_{off},Q_{delta})	The rate limiter adjusts its rate according to the two components of the ECM feedback.
Q_{mc}	Mild Congestion	ECM-Max	This message causes the maximum rate decrement of a rate limiter.
Q_{sc}	Severe Congestion	ECM(0, 0)	The rate limiter sets the rate to zero temporarily.

and Q_{delta} is the change in length of the queue since the last sampled frame. There are two cases in this situation. When the buffer utilization exceeds Q_{eq}, a positive value of Q_{off} is reported indicating the approaching of mild congestion. Otherwise, a negative value of Q_{off} is reported indicating the easing of buffer utilization after a rate cut. Corresponding rate adjustments are performed at the RP based on ECM(Q_{off}, Q_{delta}).

However, when the buffer utilization level of a CP crosses Q_{mc} but remains below Q_{sc}, the CP enters the mild congestion state. In this state, an ECM-Max message is used for notification. This message generally instructs the RP to reduce its transmission rate with a maximum cut. Finally, when the buffer utilization level of a CP crosses Q_{sc}, the CP reaches the severe congestion state. In this state, an ECM(0,0) message is used for the notification. Upon reception of this message, the RP must halt its transmission by adjusting its rate to zero for a certain predefined time period before resuming from the lowest transmission rate again.

In ECM, each rate limiter implements a variation of AIMD for its rate adjustment. The role of AIMD in a rate limiter is to regulate traffic flow from hosts to the network according to the congestion status of the network so that no excessive traffic can enter the network causing network congestion. The design of AIMD directly affects network utilization. A rate control design being too conservative may cause low utilization of networks. On the other hand, a rate control design being too aggressive may cause network over-utilization which results in network congestion and buffer overflow.

ECM employs an AIMD in a much smoother way. A rate limiter periodically increases its sending rate. If a feedback is detected, a feedback signal, F_b, is calculated using Q_{eq} and Q_{delta} by

$$F_b = -\left(Q_{off} + w \cdot Q_{delta}\right). \tag{2.1}$$

Based on the value of F_b, the rate limiter adjusts its rate, R, according to the rules

$$R \leftarrow \begin{cases} R + \min(Gi \cdot Fb \cdot Ru, \beta \cdot C), & F_b > 0 \\ R \cdot (1 - \min(Gd \cdot |Fb|, \alpha)), & F_b < 0 \end{cases}$$

where Gi and Gd are the increase gain and decrease gain respectively, Ru is the rate unit in the rate limiter, which is the granularity of the rate adjustment, and C is the capacity of the link draining the rate limiter, the quantities α and β are parameters to limit the upward and downward rate adjustments. More details on the operation of the rate limiter and its design principle can be found in [41].

2.3 Data Center Switch Control

As we have discussed in the beginning of this chapter, the aim of switch control is to ease congestion by utilizing multipath capacity to offer high bisectional bandwidth and evenly distributing traffic among these paths. Making use of multipaths and balancing load among them shall exploit redundancies of underlying topology in data centers which satisfies this purpose. Techniques for multipath routing have been a target for study in the literature. In this section we first present networking standard that enables packet switching and routing in data centers. Second, basic multipath construction and selection methods are described, which is followed by introduction of routing taxonomy. Several DCE multipath routing proposals are then reviewed. Finally, the concept of Software-Defined Networking and its enabler, OpenFlow switch, is introduced as a timely technique for switch control.

2.3.1 Standardizations

Data Center Bridging Capabilities Exchange Protocol (DCBX) defined in IEEE 802.1Qaz [7] is the basics for other networking protocols and standards, serving as a management tool for data center bridging. DCBX leverages Link Layer Discovery Protocol (LLDP) [42], [43] which is a link layer protocol that permits a network device to advertise its identity, capacity and neighbors on LAN. DCBX exchanges the capabilities of the two communicating parts to make them consistent. In case of mismatches, it will notify device manager with basic configurations. It also supports the above mechanisms such as PFC, ETS, QCN and etc.

Shortest Path Bridging (SPB) IEEE 802.1aq [44] is the replacement for STP protocols. SPB computes multiple virtual LANs (VLANs) that can share station location information. Two modes of operation are described, depending on whether the source Bridge is IEEE 802.1ad ($QinQ$) [45], which is known as SPBV, or 802.1ah ($MACinMAC$) [46], which is known as SPBM. SPBV supports a VLAN using a VLAN Identifier (VID) per node to identify the shortest path tree associated with that node. On the other hand, SPBM supports a VLAN by using one or more backbone MAC addresses to identify each node and its associated shortest path tree. Both modes use link state routing. SPBM by virtue of its $MACinMAC$ encapsulation is more suitable for a large data center than SPBV. However, a switch interface can only support a maximum of 4094 VLANs. Compared to the current data centers size which contains hundreds of thousands of servers and storage devices, this increases the computation complexity, as well as limits VLAN multipathing capability.

Equal Cost MultiPath (ECMP) [47] is a multipath routing extension for Internet routing protocols such as Open Shortest Path First (OSPF) and Routing Information Protocol (RIP). In ECMP, multipath routing is enabled when system discovers existence of two or more shortest paths of equal cost between a source-destination pair. The ECMP paths can be computed via relatively simple extensions of standard shortest path algorithms such as Dijkstra's algorithm. Once equal cost routes are discovered, load is distributed in equal proportion over ECMP paths using round-robin distribution. However, there are two major drawbacks lie in this approach: the difficulty in satisfying equal cost requirement cannot guarantee to discover multipath routes, even there exist multiple routes between each end-to-end node pair; moreover, the static evenly distribution of traffic load, taking no account of network congestion status, may continually assign traffic to an already congested route. RFC 2991 [48] and RFC 2992 [49] discuss multipath routing in general.

Transparent Interconnection of Lots of Links (TRILL) is an IETF standard implemented by devices called Routing Bridges (RBridges) or TRILL switches. RBridges [50] are proposed in 2004 which run a link state protocol amongst themselves to broadcast connectivity to all other RBridges to compute pairwise optimal paths, and generate distribution trees to deliver frames. Later based on RBridges, TRILL standard protocol mitigates the loop problem by using a Time To Live (TTL) field in the frame header, and deploys multipaths to support scalability. The major drawback

of TRILL is that it only allows ECMPs to deliver frames. When TRILL is deployed in a typical layered data center as shown in Fig. 2.2, the influence of such a limitation is minimal. However, when TRILL is running in a network with an arbitrary topology, its performance may be adversely affected by this limitation because an arbitrary topology cannot guarantee the availability of ECMPs between two RBridges.

2.3.2 Multipath Routing Techniques

Multipath routing aims to exploit the capacity of underlying physical network by providing alternative paths between any pair of end nodes. There are two parts regarding a multipath routing algorithm: construction of multipaths given a network topology and partition of traffic load based on a set of multipaths.

2.3.2.1 Multipath Construction

To find the shortest path, Dijkstra [51] algorithm is widely accepted and commonly used. Its idea has been extended for construction of K-th shortest path [52], [53], [54]. In data centers, multipaths should be constructed in a similar way. Here, there are generally two typical methods for paths construction, namely generating algorithms [55], [53] and reducing algorithms [56], [54].

Generating algorithms generates a tree from the shortest path. They build a pseudo-tree based on K shortest paths, which first finds the shortest path, then generates the second shortest path upon the first shortest path iteratively until K paths are determined. Reducing algorithms, on the other hand, build subset of current topology repeatedly by excluding the current shortest path. They compute K shortest paths by finding the shortest path in the current topology, then removing the path and finding the shortest path in the remaining topology continuously. In our switch control scheme proposed in Chapter 4, we shall assume that multipaths are established from a particular algorithm mentioned above. Our focus there will be the routing of packets over the established multipaths.

2.3.2.2 Traffic Routing

Routing protocols communicate network information and determine how packets would be routed. To yield a higher bandwidth utilization or shorter delay, various routing protocols that have been proposed for improving STP or multipath routing.

Several improvement or extension attempts of STP, including Rapid Spanning Tree Protocol (RSTP) [19], Multiple Spanning Tree Protocol (MSTP) [57], [17], Per-Virtual LAN (VLAN) Spanning Tree (PVST) [17] and Rapid Per-VLAN Spanning Tree (RPVST), have been carried out. RSTP speeds up convergence time following a link failure, but it stills uses a single tree to delivery traffic between communication pairs. PVST improves link utilization by deploying spanning tree in each VLAN. Since PVST treats each VLAN as a separate network, it has the ability to balance traffic load at Layer-2 by forwarding the traffic of some VLANs on one trunk and the traffic of other VLANs on another trunk without causing a loop. However, since VLANs are separated subnets, a router or Layer-3 switch is required to communicate devices in different VLANs. Hence the VLAN method still inherits many challenges of an IP network. Extension of PVST, known as Multiple-VLAN Spanning Tree (MVST), allows similarly connected VLANs to be grouped into a single spanning tree and thus makes the structure more scalable. Further, Viking design [58] uses MSTP for faster fault recovery and deploys a central manager to dynamically select trees according to its global information on traffic load.

Another multipath scheme, named OSPF-Optimized MultiPath (OSPF-OMP) [59], is an improved version of ECMP for OSPF routing protocol. In multipath construction, OMP finds multipath routes by simple extensions of shortest path routing algorithms such as Dijiskra algorithm. In traffic partition, OMP implements proportional traffic distribution. It does not require these routes to be of equal cost and distributes traffic over multipath routes in inverse proportion to the path utilizations. Unlike ECMP, OMP ensures that packets belonging to a flow are always forwarded on the same path. Thus, traffic is distributed at flow granularity, which avoids the out-of-order packet problem of ECMP. However, OMP adjusting decisions are made by nodes based on path utilization information, multiple nodes may simultaneously assign traffic to several multipath routes sharing common links. Although congestion at the original path may be avoided, other paths may become congested as its side effect. This could lead to network instability of multiple nodes switching traffic back and forth.

Besides, there are other research directions. Multiple Path Algorithm (MPA) [60] calculates a subset of loop-free paths. Other proposals aim to minimize delays such as Discount Shortest Path Algorithm (DSPA), or maximize throughput such as Capacity

Removal Algorithm (CRA) [61]. Moreover, Vutukury *et al.* argue the demand for loop-free network even in a transient instant with their proposal of distant routing algorithms of Multipath Distance Vector Algorithm (MDVA) [62] and link state algorithms of Multipath Partial Dissemination Algorithm (MPDA) [63]. In addition, proposals such as [64], [65], [66] focus on providing transport layer multipath routing algorithms.

The multipath traffic routing and distribution scheme proposed in Chapter 4 is a source routing protocol which access switches accumulate link utilization statistics and make routing decisions on a flow granularity. Together with a congestion rate control scheme, the proposal not only ensures a load balancing network but maintains traffic loads at a mild level.

2.3.2.3 Taxonomy of Routing Algorithms

In this section, we introduce the taxonomy of routing algorithms, which are categorized into two sets depending on how they select among the set of available paths between each pairwise source-destination nodes.

Oblivious algorithms select a path without consideration of any information other than the source-destination network addresses or flow ID. That means, routing decisions are oblivious to the network's present state. For example, there may exist several shortest paths between the source and destination and the routing algorithm distributes uniformly the communication load such as ECMP.

In this category, a special subset, **deterministic routing algorithms**, is algorithms that always select the same path for given pair of source and destination nodes, even if there are multiple possible paths. These algorithms ignore path diversity of underlying topology and thus perform poorly in load balancing. Despite this, they are used in most commercially available machines because they perform well under uniform traffic as well as they are easy to be made deadlock-free in meshes and implemented in hardware. STP is an example of Deterministic Algorithms.

Adaptive algorithms, on the contrary, adapt to the state of the network and make routing decisions based on network state. The information may include status of the node, link utilizations and queue length of switches' buffers and traffic loads. More examples will be discussed next in Section 2.3.3.

2.3.3 Existing Proposals

Most recent proposals about Layer-2 forwarding protocols focus on eliminating the restrictions of using a single spanning tree and utilizing multipaths for traffic delivery instead. Here, we present these proposals according to the taxonomy introduced in the previous Section 2.3.2.2.

2.3.3.1 Oblivious Routing

A number of data center routing protocols are oblivious routing due to its simplicity for deployment, among which we enumerate some of the important proposals in this part.

SEATTLE employs a one-hop Distributed Hash Table (DHT), mapping a host's location with its MAC address, and then delivers the frame along the shortest path using the location acquired. SEATTLE admits routing loops for unicast traffic and proposes a new group solution for broadcast/multicast traffic. Within a group, it runs over a single spanning tree protocol, eliminating the possibility of loops for such traffic. However, such a mechanism also limits the traffic to be delivered over a single path, instead of multipath, which is critical for scalability and better utilization of network resources.

The main tradeoff with oblivious routing is between locality and load balance. One algorithm focuses on locality is **Fat Tree** [3], as discussed earlier. In this proposal, it creates a concept of two level lookups to assist with multipath routing across Fat Tree. With the introduction of two level routing table, Fat Tree differentiates intra-pod traffic and inter-pod traffic, which ensures minimal routing (always follow one of the shortest paths). Specifically, the mechanism first filters a packet by routing it to a lower level switch/host (within the subnet) if the destination address of the packet matches the first level entries of the routing table. Otherwise, the packet will be distributed to a higher level switch based on the second level entries. In their design, the generation of routing tables follows a static manner by diffusing traffic on multipaths based on the suffix of its destination address. This algorithm performs well for uniform traffic pattern, however, for other traffic patterns, especially permutation traffic [1], it can hardly achieve the maximal network capacity.

[1] As defined in [24], permutation traffic describes such a traffic pattern in which each source sends all of its traffic to a single destination. It is often used to stress a network.

Valiant Load Balancing (VLB), on the other hand, performs well on load balancing. It guarantees equally spread the load of any traffic pattern, by bouncing individual packets from a source switch in the mesh network through randomly chosen intermediate switches, which finally forward those packets to their destination switch. Albert Greenberg *et al.* used VLB in Monsoon network [67], which consists a Layer-3 ECMP portion, connecting the data center to Internet, and a Layer-2 mesh portion, connecting all the servers to Layer-3. In the design of Monsoon, a server can serve as an intermediate load balancer before routing a packet to its final destination. Recent realizations of VL2 [68] perform randomized forwarding on a per-flow rather than on a per-packet basis to preserve packet order. However, VLB achieves load balancing by compromising its locality. That means, even traffic between near hosts gives no better performance than worst case traffic.

2.3.3.2 Adaptive Routing

Instead of picking an algorithm independent of the traffic pattern as in oblivious algorithm, we can imagine an algorithm that adapts to current traffic conditions probably performs better. This simple idea forms the basis of adaptive routing algorithms.

As proposed in [69], **Adaptive Routing** takes network congestion status into consideration. In the design, a switch lists multipath interfaces to each destination with priority. These interfaces snoop and keep track of current level of congestions on the path. Based on the information acquired, routing algorithm selects the best available interface.

Hedera is proposed [70] as another adaptive solution based on PortLand testbed to track the current levels of congestion on all available paths between source-destination pairs and then to maximize bisectional bandwidth with minimal rescheduling on active flows. In particular, Hedera implements a central network fabric manager to track the behavior of large flows and leverage decisions on scheduling among them.

2.3.4 Software-Defined Networking

A well-designed adaptive routing algorithm theoretically should outperform an oblivious routing algorithm, since it includes network state information not available to the oblivious algorithm. However, many adaptive routing algorithms yield poor worst-case performance. This is largely due to the local nature of most practical adaptive routing

algorithms. Because they use only local network state information, for example local queue lengths, in making routing decisions, they route in a manner that balances local load but often results in global imbalance. Other efforts attempting to obtain global information through feedback mechanisms often lead to an overhead that congests network even further.

The introduction of a concept called Software-Defined Networking (SDN) creates an opportunity to achieve the best of both worlds. The concept of SDN comes from a concept called Software Defined Radio, which modulates/demodulates using software instead of hardware as in Cognitive Radio. Similarly, different from traditional network which data plane and control plane are implemented on hardware, SDN decouples intelligence that controls the flow to a distinct software application and handles packet traffic still on hardware. This separation opens up the control flow to be easily managed and remotely accessed. Employing a central controller, which amasses knowledge of switches' routing information and network status, SDN manages routing algorithms separately while keeps data flow running on original network paths. Hence, SDN provides much easier modification to switch algorithms, as well as faster control over network status changes.

OpenFlow project, as the most popular enabler for SDN, is proposed by McKeown *et al.* [71]. In this platform, a small piece of OpenFlow firmware (software embedded in hardware) is installed on switches such that engineers can access flow tables, rules for routing traffic. Moreover, this feature presents OpenFlow as an ideal tool for academic and research institutions to obtain network statistics and explore network performance as well as innovative algorithms, while protecting the proprietary routing instructions that differentiate one company's hardware from another. Many vendors such as Cisco, Hewlett-Packard have announced their intention to support OpenFlow, thus making it a step further to industry availability.

In our research in Chapter 4, we also employ SDN concept to offer a quick exchange of network status and routing information. This technology helps avoid the performance degradation due to worst-case traffic pattern and obtain overall control of network. The proposal can be implemented on OpenFlow platform or other implementations of SDN technology.

2.4 Data Center Link Control

The objective of link control is to adapt to the congestion status with link assignment proportionally which assists to achieve power proportional networking. We first present an Ethernet link control standard, then introduce internet techniques for power-efficient link control. Thirdly, two data center power control solutions are discussed.

2.4.1 Standardizations

With regard to link power control, there is no mechanism standardized in data center, but IEEE 802.3az [72] specifies a standard for green Ethernet technology.

Energy Efficient Ethernet (EEE) is a set of enhancements of Ethernet networking standards. EEE allows lower power consumption while system is idle or operating at a light load. The initiative of this working group is to reduce network idle state power consumption, since energy is uniformly consumed to keep the physical layer transmitters on all the time. Motivated by reducing energy cost, the task group switches data links into sleep mode using a low-power-idle (LPI) that could signal for a suspension of the transmit chips in the system within a specified time. This mechanism only triggers an on/off switch according to system traffic load, yet it lacks of a fine-grained tuning. Moreover, originally designed for LAN, EEE takes no account of reducing power consumption with regard to data center redundancies.

2.4.2 Internet Link Control Techniques

In this part, we describe existing work on reducing power consumption of LAN switches, Ethernet network links, and NICs.

One of the earliest research on dynamic link power management of wired computer networks is given by Gupta *et al.* [73]. They propose two solutions to reduce the power consumption of wired networks. First, they suggest to power off network interfaces in LAN switches during packet inter-arrival times. It is shown that by adding extra memory buffers at the interfaces, packet loss can be minimized. Second, they propose that routing protocols can be modified so that all traffic on parallel links are aggregated into one link during low-traffic periods in order to allow idle network links be powered off.

In fact, some products conforming to newer Ethernet over UTP standards still maintain backward compatibility. Thus they are capable of operating at lower link data rates which attracts some research that focuses on switching link rate to a lowest adequate rate depending on traffic load.

The main concern of the aforementioned method challenges the feasibility of rate negotiation. One potential solution considers to use Auto-Negotiation [74] mechanism, a procedure to select common transmission parameters such as speed, duplex mode, and flow control when connecting two devices. However, using Auto-Negotiation for switching link data rate would disable the link for hundreds of milliseconds. This is not feasible and may even result in packet loss in switches.

Subsequent proposal of Adaptive Link Rate (ALR) work [75] discusses similar ideas adapting ethernet links with traffic load at different rates, and evaluates them with an analytic model and simulation using synthetic traffic patterns. With a quick negotiation, this proposal reduces the switching time to around 1ms, which is 100 times shorter than using Auto-negotiation.

2.4.3 Existing Proposals

Large-scale interconnection network has been a fertile area of research, however, the notion of energy proportional networks in large data centers still remains an important and unexplored topic.

Energy efficient data center research has different focuses. Some of the research focuses on a more energy-efficient communication by optimizing the physical characteristics of the link. Heller *et al.*, recently proposed ElasticTree [76] to save energy in data centers by continuously monitoring data center traffic conditions at the system scale, and then determining which parts of the fabric to be activated and powered off the rest.

Others focus on providing energy efficient topologies. Abts *et al.* [22] addresses using flattened butterfly topology with control on each individual direction of the link can provide more efficient power usage.

As representatives of the two major approaches, a close investigation of these two proposals is presented in the following.

2.4.3.1 ElasticTree

Heller *et al.* proposed power proportional data center network in 2010 [76]. In a power proportional data center, the power consumption increases as traffic load becomes heavy. ElasticTree achieves this goal by disabling part of the network devices. Specifically, it employs a centric optimizer to calculate the minimum power network subset based on the network topology, traffic matrix and switch power models. Through this centric optimizer, it can dynamically adjust the set of active network components, including links and switches, given data center traffic load. This work provides a framework for data center energy saving, as well as a performance baseline for a certain network settings, based on Fat Tree topology.

There are three optimizer models described in the proposal, including *formal model*, *Greedy Bin-Packing* and *Topology-aware Heuristic*. The formal model is presented as the optimal solution, which yields a subset of the original topology and the routes taken by each flow to satisfy traffic matrix, with the time growth of $O(n^{3.5})$. The other two models are heuristic: Greedy Bin-Packing evaluates possible paths from left of network topology for each flow, and selects first path which offers sufficient capacity. The inherent problem with this approach is that it may fail to find a solution for all flows. Topology-aware Heuristic utilizes the equality of the number of required switches in the aggregation layer and the number of links required to support incoming traffic. Similar calculation applies between pods and the cores. But this approach is under the assumption that the flows are perfectly divisible that reduces its feasibility.

Apart from the aforementioned algorithms in traffic distribution, this work does not take link rate into consideration. Besides in the experiments, results are all based on preset traffic matrix, which means that the system have a complete knowledge of incoming traffic. However, this is unrealistic in practice. Though the paper also considers setting safety margin, which reserves some amount of capacity, traffic prediction and also fault tolerance to further complete the solution, most of them remain as framework with not much detail provided and there is still room for further exploration.

2.4.3.2 Energy Proportional Data Center Networks

Compared to ElasticTree which mainly focuses on Fat Tree topology, [22] presents that Fat Tree topology inherently consumes higher energy and suggests with a lot switches

for redundancies using a flattened butterfly topology, details provided in Section 2.1.2.1, can yield a pronounced saving.

In their work, the authors argue that a flattened butterfly topology consumes less power at full utilization for fewer optical transceivers and switching chips it requires, but also the proposal offers more dynamic range of network power. Specifically, the mechanism proposed is designed to control each unidirectional channels independently, which effectively curtails energy usage especially when traffic pattern follows asymmetric distribution. In fact, based on our observation, such type of a traffic pattern often appears in data center environment. This fine grained control over links explores the system potential flexibility on power consumption. However, after examination of several switch models, we find that they still lack separate control towards each direction. More hardware supports are needed to implement this design. Moreover, the topology of flattened butterfly saves power but at odds with deployment costs and path lengths, which is prone to increase transmission delay.

2.5 Summary

In this chapter, an in-depth literature review has been provided from four perspectives. Firstly, data center topology serves as the foundation of all proposals. We reviewed standard topology as well as emerging topologies, among which tree structure based proposals are seen as a prospective transition from standard topology to next generation scalable data center topology. Our proposed schemes in the following three chapters are all based on tree topologies. Secondly, source control standard, techniques and existing proposals have been reviewed. A combination of AQM and AIMD techniques has been found as proper tools for prioritized source control. In Chapter 3, a source control scheme for throttling congestion is developed from AQM and AIMD combination. Thirdly, switch control standard, techniques, existing proposals and SDN technology have been covered. In this part, SDN technology has been seen as a useful way to adopt adaptive routing algorithm without suffering much delay due to information update. In Chapter 4, a switch control scheme for easing congestion is proposed using SDN to yield high throughput in data centers. Lastly, link control standard, techniques, and existing proposals are reviewed. In this part, ALR has caught the attention for adaptively changing link rate. In Chapter 5, a link control scheme for adapting congestion is

proposed with consideration of both link rate and load distribution according to traffic volume, thus it saves energy consumed in data centers. Combining the three schemes in the following chapters, a complete congestion management solution is proposed.

Chapter 3

Source Control for Throttling Congestion

3.1 Motivation

In Chapter 2, we discuss that source control can easily adjust and differentiate source sending rates. Through source control, two requirements of congestion management can be addressed, namely lossless congestion control and priority differentiation. To be specific, lossless congestion control ensures Ethernet to carry FC and IPC traffic with no frame drop by adjusting source rates; priority differentiation protects FC and IPC traffic with higher bandwidth and shorter latency by differentiating source rates.

Several separate efforts have been made towards these two requirements, including DCTCP and ECM for end-to-end congestion control as well as PFC and ETS for priority control as reviewed in Chapter 2. From our observation, the current end-to-end congestion control proposals that hold excessive traffic from entering network core can combat congestion sufficiently, but priority differentiation still remains at a coarse-grained level. Intrigued by the demand for a fine-grained priority differentiation, we propose a source control scheme for prioritized congestion control. The scheme consists two major parts, *active queue management* and *rate limiting algorithm*. The active queue management consistently inspects buffer levels of network switches. When any buffer level exceeds certain thresholds, feedback is sent to the sources. Upon receiving congestion feedback, the rate limiting algorithm throttles source sending rates to prevent excessive traffic admission. Using separate sets of AIMD parameters, this

algorithm adjusts source rates respectively for different priorities. Thus, our innovative source control scheme not only tunes throughput on various priorities precisely, but also avoids network congestion at the same time.

Our proposal mainly involves with two mature technologies, namely AIMD and AQM, described in Section 2.2.2. The combination of AIMD and AQM has been proven to be a viable way for congestion avoidance [77], [78], [79] and it is currently considered as a strong candidate for Ethernet congestion control in DCE. One recent proposal ECM, details can be found in Section 2.2.3.2, is based on AIMD and AQM operations to achieve traffic congestion control for lossless traffic. However, the lack of differentiation on prioritized traffic has made their congestion control scheme inadequate for DCE to deal with a mix of data center hybrid traffic.

In this chapter, we provide details of our source control scheme that supports differentiated congestion control for prioritized traffic. Through analytical and simulation studies, we demonstrate the potential of the proposal for differentiated congestion control in Ethernet. The rest of this chapter is organized as follows. Section 3.2 presents the design of our proposed source control scheme. Section 3.3 provides control theoretic analytical model, which focuses on computation of differentiation ratios and recommend sample probability. Section 3.4 presents simulation results covering studies in system design and parameter design, and comparison with other data center congestion control schemes. Finally, Section 3.5 summarizes this chapter. This research work has been summarized and published in [80], [81].

3.2 Differentiated Congestion Control

Given the wide application of AIMD rate controller and AQM in the Internet congestion control, this approach represents a potential candidate for congestion control in DCE design. However, due to the consolidation of FC, IPC and Ethernet, Ethernet will face coexistence of the traditional LAN traffic, SAN traffic and HPC traffic [82]. Preparing for the handling of a mix of traffic with different characteristics, we argue the need for differentiated congestion control handling. To address this issue, we propose using different AIMD parameter sets for the rate limiters to achieve congestion control differentiation, and show its performance feasibility for this design. Our proposed mechanism is illustrated in Fig. 3.1. This model utilizes two components to regulate

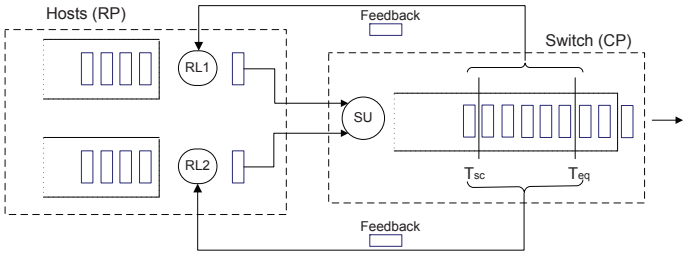


Figure 3.1: System model of Differentiated Congestion Control.

traffic, queue management and rate limiters. Queue management in the congestion point (CP) helps detect a potential congestion, while rate limiters in the reaction point side contribute to the regulation of traffic flow to prevent congestions.

In general, our proposed mechanism can easily satisfy an arbitrary number of traffic priorities for different congestion control handling. Focusing on data center application, we shall primarily focus on congestion differentiation between LAN and SAN traffic.

3.2.1 Queue Management in Congestion Point

A CP features a queue management which is mainly responsible for congestion detection, congestion notification and frame drop policy. For congestion detection, similar to ECM, we propose a sampling function to control the frequency of buffer inspection. In the sampling function, incoming frames are being sampled on a byte arrival basis. That means, the CP samples a frame every fixed number of bytes received, and this fixed number of bytes is calculated by length specified $E[L]/p_s$ where $E[L]$ is the average frame length and p_s is a preset sampling probability. Whenever a frame is sampled, the CP also inspects the buffer status to detect congestion events.

The congestion notification is performed by a feedback function. When a frame is sampled and the buffer is inspected, the CP may notify the source whose frame is sampled to adjust its rate based on the buffer status. In our design, there are two thresholds, namely T_{eq} and T_{sc}, representing an equilibrium congestion level and a severe congestion level, respectively. To simplify the operation, we exclude the use of mild congestion control in our design. The consequent of excluding mild congestion control is the increase sensitivity to a sudden upward change of traffic arrivals. In our design, this sensitivity may be offset by a higher sampling probability. During a buffer inspection, if the buffer utilization exceeds T_{eq}, a notification will be sent to

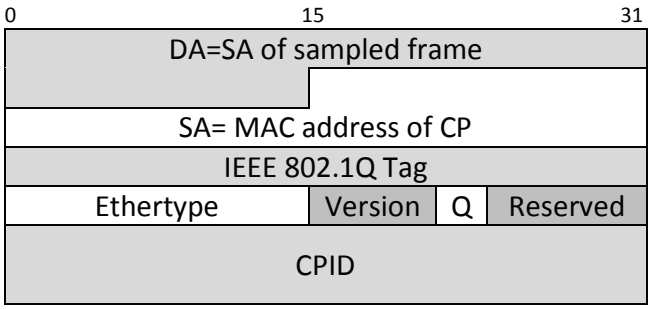


Figure 3.2: Key fields of feedback message.

the source whose frame is sampled. This notification instructs the rate limiter of the source to adjust its rate downward according to the AIMD parameters. If the buffer utilization exceeds T_{sc} indicating the appearance of severe congestion, a notification is sent instructing a source to halt a transmission for a predefined period of time, and the source shall resume its transmission at the lowest predefined rate.

Figure 3.2 shows the key fields of feedback frame. DA is the source address of the sampled frame, and SA is the address of the port in the switch that samples the frames. The message type can be identified as a notification, whether is to slow down or to stop the transmission, by the EtherType field. CPID is ID of the CP. The remaining fields follow IEEE802.1Q standard, such as IEEE802.1Q Tag, Version, and the field of Q.

The CP also implements a frame drop policy to achieve differentiation in traffic congestion management if congestion persists. In our design, when the buffer utilization exceeds T_{sc}, all LAN traffic, being the low priority traffic, will be dropped until the buffer level returns below the equilibrium threshold, T_{eq}. This frame drop policy further protects the SAN traffic during severe congestion. Examples of such a situation is presented later in Section 3.4.2

3.2.2 Rate Limiters in the Reaction Point

A reaction point (RP) on the host side implements rate limiters to regulate traffic flows. A rate limiter regulates the traffic flow by controlling the transmission rate using an AIMD based rate adjustment operation, and a CP provides necessary feedback for the AIMD operation to perform rate adjustment. We propose using different sets of AIMD

parameters to achieve differentiated congestion handling of different types of traffic. In the following, we shall describe the AIMD operation related to our design.

Each source consists of an AIMD-based rate limiter. Similar to the mechanism employed in TCP, the rate limiter maintains a variable called *congestion window* to regulate the transmission rate. The value of congestion window is initialized to be one. This value increases linearly over a predefined constant time interval called a *slot*. Congestion window specifies the number of frames that a source can transmit at the beginning of each slot. As the value of congestion window increases, the number of frames that a source can transmit into the network increases accordingly. This increase in frames into the network may cause network congestion. At the CP, each buffer in the Ethernet switch executes the above mentioned ECN to monitor the buffer utilization and notify the source to regulate its transmission rate by adjusting the congestion window.

When a notification of rate cut is received, the source immediately reduces the value of its congestion window by a certain percentage. This action directly reduces the number of frames the source can transmit to the network and eases the network congestion. After that, the congestion window continues to increase for every slot time until the next notification of rate cut appears. Let $x_i(t)$ be the load source i transmits to the network at time t, source i will adjust its load at the next slot by

$$x_i\left(t+1\right) = \begin{cases} a_i + x_i\left(t\right), & X\left(t\right) \leq T_{eq} \\ b_i x_i\left(t\right), & \text{otherwise} \end{cases} \tag{3.1}$$

where $X(t) = \sum_i x_i(t)$ is the sum of the number of frames transmitted by all sources. Note that AIMD can be described by two parameters, which are a and b where $a > 0, 0 \leq b < 1$. This gives rise to several strategies of congestion control. In the real implementation, a parameter called *step of a* is also used, as the unit of transmission rate, denoted as u. The real increase rate comes from the window size multiplying the step, namely $a \cdot u$.

3.3 Analysis

The selection of parameters for rate limiter design is critical to ensure that the targeted differentiated service is achieved while maintaining a high performance of operation.

In our preliminary work in [80], we demonstrated AIMD operation for service differentiation specified by bandwidth differentiation ratio between two classes of traffic types. We introduced a semi-Markov process to model the AIMD operation where the model computes bandwidth differentiation ratios with various AIMD settings. The use of semi-Markov process, however, is limited to the study of a small number of traffic types due to scalability of the model.

In this chapter, by taking a different approach, we introduce a control theoretic analytical model for the study of our proposed differentiated congestion control mechanism. This model enables the study of bandwidth differentiation ratios for an arbitrary number of traffic types and the description of marking probabilities for system stability analysis.

3.3.1 The Control Theoretic Model [1]

In [1], Hollot *et al.* propose a control theoretic analytical model for the performance study of TCP congestion control operation. Focusing on the equilibrium state of the system, a set of differential equations are constructed to describe the AIMD operation of TCP. Following the same approach, we model our proposed differentiated congestion control solution as a control system described in Fig. 3.3. As our system includes multiple rate limiters with different AIMD parameter settings connecting to a common switch, we extend the model presented in [1] from a single AIMD type to multiple AIMD types resulting several AIMD flow control blocks connecting to a single bottleneck as shown in Fig. 3.3. With this extension, the formulation in (1) given in [1] is modified accordingly from a single equation to a set of simultaneous equations as

$$\dot{W}_i(t) = \frac{a_i}{R(t)} - \frac{(1-b_i)\cdot W(t)W(t-R(t))}{R(t-R(t))} \cdot p(t-R(t)) \tag{3.2}$$

and

$$\dot{q}(t) = \sum_{i=1}^{n} \frac{W(t)}{R(t)} N_i(t) - C \tag{3.3}$$

where a_i, b_i describe the AIMD parameters as defined in Section 3.2, $\dot{x}$ denotes the time-derivative of x, and

$$\begin{aligned} W_i &\doteq \text{expected window size of link } i \text{ (frames)}; \\ q &\doteq \text{expected buffer level (frames)}; \\ R &\doteq \text{round-trip time (secs)}; \\ C &\doteq \text{network capacity (frames/sec)}; \\ N_i &\doteq \text{link number of link type } i \text{ (links)}; \\ p_i &\doteq \text{marking probability of link type } i. \end{aligned}$$

Similar to TCP that employs AIMD congestion control, our differentiated congestion control scheme also uses AIMD congestion control achieving stability in operation. As a result, the developed control theoretic model gives a steady-state solution, and the solution can be obtained by setting $\dot{W}_i = 0$ [1], [83]. In the steady-state operation, applying $\dot{W}_i = 0$ for all i, we have

$$\hat{W}_i^2 = \frac{a_i}{(1-b_i)p_i} \tag{3.4}$$

where $\hat{W}_i$ denotes the steady-state solution of $W_i(t)$.

3.3.2 Application of Control Theoretic Model for Multiple Traffic Flows and Types

In our design, the quantities of marking probabilities p_i have the same value. Consequently with j types of AIMD settings each with N_j flows, we obtain

$$\hat{W}_1 : \hat{W}_2 : \cdots : \hat{W}_j = \sqrt{\tfrac{a_1}{1-b_1}} : \sqrt{\tfrac{a_2}{1-b_2}} : \cdots : \sqrt{\tfrac{a_j}{1-b_j}} \tag{3.5}$$

which gives relationship between AIMD parameter settings and bandwidth utilization ratio. This relationship provides an AIMD parameter design guideline to achieve bandwidth differentiation among all j types of traffic.

Besides, in the steady-state, the aggregated utilization is bounded by C, thus we have $\sum_i \hat{W}_i \leq C$. With Eqn. (3.4), we yield

$$\sum_{i=1}^{j} N_i \sqrt{\frac{a_i}{p_s(1-b_i)}} \leq C \tag{3.6}$$

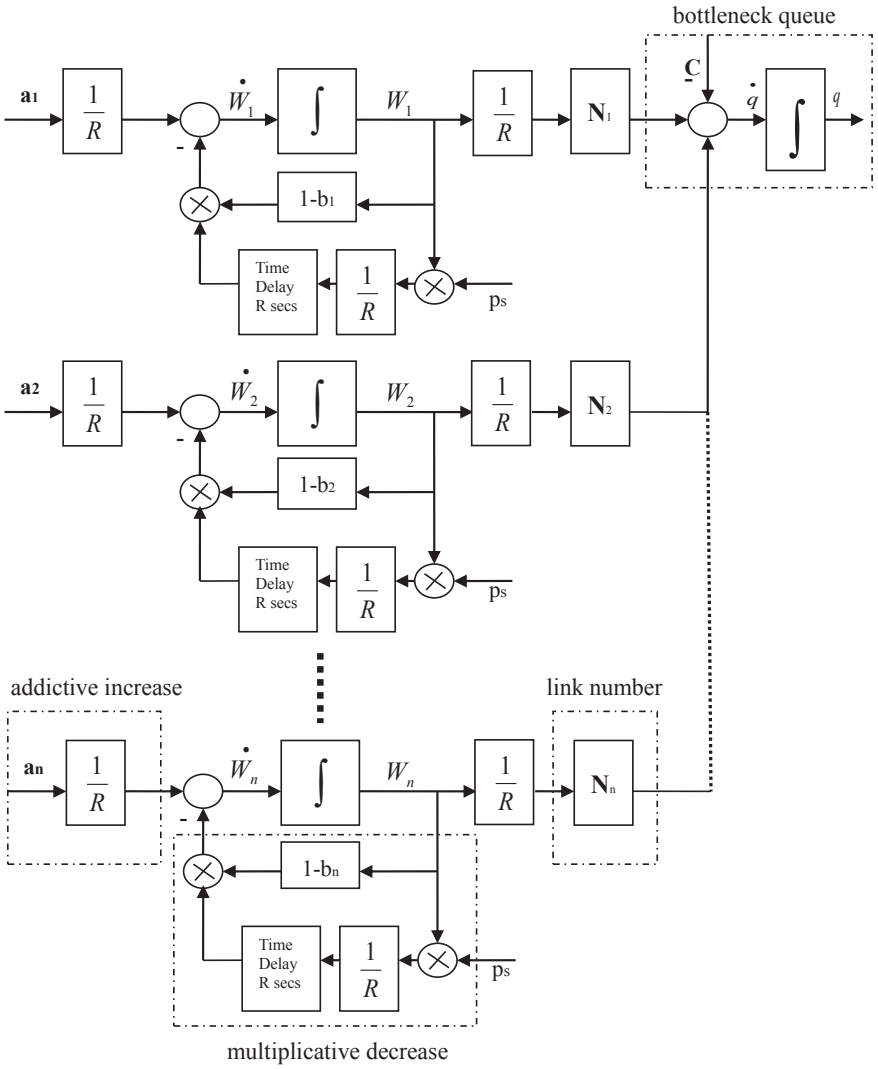


Figure 3.3: Block diagram of Differentiated Congestion Control flow-control model.

where p_s is the common marking probability for all traffic types.

With a targeted bandwidth utilization ratio, Eqn. (3.5) provides solutions for AIMD settings, or a_i and b_i for all i. Using the obtained a_i and b_i values and Eqn. (3.6), we can find a marking probability for the system such that the aggregated send rate of all rate limiters matches the link utilization by simply solving $\sum_i \hat{W}_i = C$.

In our system, the marking probability directly governs the frequency of sampling of the buffer status and incoming frames. In other words, the marking probability affects the timeliness of the detection and reaction on a congestion event, and the choice of

this setting is critical to ensure the effectiveness and stability of the congestion control. If the marking probability is set too high, the system may overreact to a congestion event by issuing excessive feedback messages. Conversely, if the marking probability is set too low, the system may be slow in reacting to a congestion event making the congestion recovery difficult. Consequently, an optimal setting of marking probability not only achieves full utilization of the link, but also ensures the adequate reaction of a congestion event so as to achieve stable control.

In our design, the congestion point sends no feedback when the buffer level is less than the equilibrium threshold T_{eq} as the buffer is underutilized. In our model, this is equivalent to setting the marking probability to zero. When the buffer level crosses T_{eq} but not the severe congestion threshold T_{sc}, a constant marking probability p_s applies. In this case, each incoming frame is sampled along with the buffer level with a constant probability of p_s. When the buffer level crosses T_{sc}, the congestion point sends feedback to the source to halt its transmission immediately. In our model, we set marking probability to one to describe the halt of a transmission. This handling is adequate as when the marking probability is set to one, all transmitted frames will trigger a feedback message instructing a downward rate adjustment. A series of these messages will quickly bring the congestion window to zero which is equivalent to halting a transmission. The above discussion can be presented in the following as

$$p = \begin{cases} 0, & q < T_{eq} \\ p_s, & T_{eq} \leq q < T_{sc} \\ 1, & q \geq T_{sc} \end{cases} . \tag{3.7}$$

Since the objective of the design is to maintain the buffer level $q(t)$ to operate in the range $[T_{eq}, T_{sc}]$ at any time t, we shall focus on the performance under the condition where $T_{eq} < q(t) < T_{sc}$. In our design, $q(t)$ is forced to return back into the range $[T_{eq}, T_{sc}]$ when it falls outside the range. Specifically, when $q(t) < T_{eq}$, each rate limiter increases its congestion window to bring $q(t)$ up above T_{eq}. Likewise, when $q(t) > T_{sc}$, a collection of rate limiters is notified to halt their transmissions to quickly bring $q(t)$ back below T_{sc}.

3.4 Experiments and Result Discussions

Performance studies of our proposed differentiated congestion control scheme are conducted. We perform a number of experiments to demonstrate the effectiveness and

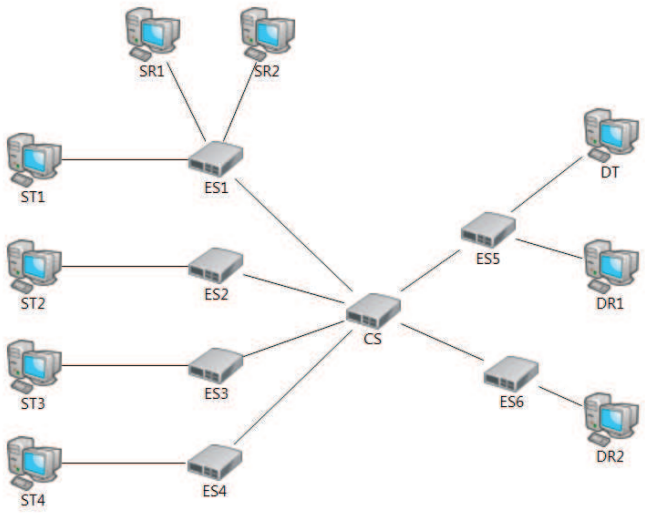


Figure 3.4: Network topology for OMNET++ simulation study.

the performance benefits of our proposed scheme. In all the presented experiments, we also present simulation results to validate our numerical analysis. Our simulation is implemented by OMNET++ 4.0 simulator [84]. In our rate limiters, the parameter u describing the step of a is set to 40Mbps. We also set the initial send rate to 40Mbps.

In our experiments, we follow the setup given in [85] and test the performance of our proposed scheme. Figure 3.4 depicts the network topology used in our experiments. The data rate for all links is 1Gbps. In our scenario, the two sources SR_1 and SR_2 generate frames to the two destinations DR_1 and DR_2 at a certain rate. The two sources carry different types of traffic, where SR_1 carries the SAN and SR_2 carries the LAN traffic. Their transmission rates are regulated by rate limiters with SR_1 implementing AIMD(a_1, b_1) and SR_2 implementing AIMD(a_2, b_2)[1].

In Experiments 2-4, we include background traffic in the experiments. We use the nodes from ST_1 to ST_4 to generate Constant BitRate (CBR) background traffic destined to the node DT through end switches ES_1 to ES_4. The rate of each CBR traffic is set to 4.8Mbps. As they are background traffic, they do not implement a rate limiter.

Given that all traffic flows share the common core switch CS , it represents the

[1] We use the notation of AIMD(a, b) to describe a rate limiter implementing AIMD with parameters a and b.

bottleneck of the network. The bottleneck may be due to lack of bandwidth capacity or limited switch processing capacity. In any case, this causes the buildup of buffer in the CS and activates the congestion control.

We apply heavy load into the network to test the performance of the congestion control in the bottleneck CS with the processing rate of 20,000 frames per second. For the frame length of 1500 bytes, this processing rate is equivalent to 240Mbps, which represents the bottleneck of the network. In the core switch, the thresholds T_{eq} and T_{sc} are set to 50,000 and 100,000 bytes respectively.

To give a comprehensive illustration of the effectiveness and the performance benefits of our proposed mechanism, we have conducted five experiments. We shall report the experiments and results in the following subsections.

3.4.1 Experiment 1: Performance Comparison with Various Schemes

In the first experiment, we compare our scheme with normal Ethernet, ECM and DCTCP. The objective of this experiment is to investigate the effectiveness and system stability of our congestion control scheme.

We compare five cases where in the first three cases, we consider standard Ethernet without congestion control, ECM and DCTCP respectively. To compare the performance with these schemes without the capability of differentiated congestion control, in the fourth case called DCC1, we reduce our scheme to an undifferentiated congestion control by simply using a single AIMD parameter set in the system. In this setup, AIMD parameters in both sources are set to AIMD$(1, 0.5)$. Finally, in the fifth case called DCC2, we consider a scenario requiring a differentiated congestion control with coexistence of SAN and LAN traffic, where AIMD$(1.5, 0.8)$ is set for SAN and AIMD$(1, 0.5)$ is set for LAN. In order to focus on the influence of the various congestion control schemes on the performance, we do not include background traffic in this experiment.

The parameter setting for this experiment is shown in Table 3.1. The detailed description of the parameters for various schemes can be found in [41], [40]. We show the throughput of SR_1 and SR_2 separately and the total throughput of the two sources, in Megabyte (MB), in Table 3.2. In the first case, the pure Ethernet switch performs its best effort to forward data frames. We can see that there is no differentiation in the bandwidth utilization between the two sources. Moreover, since we set a data frame

processing time of 0.05ms for the core switch, the switch represents the congestion point in the network. With no control on the source, excessive frames generated by the two sources run freely to the core switch causing serious congestion and resulting high drop rate. In this test, almost half of the arriving traffic is discarded due to buffer overflow.

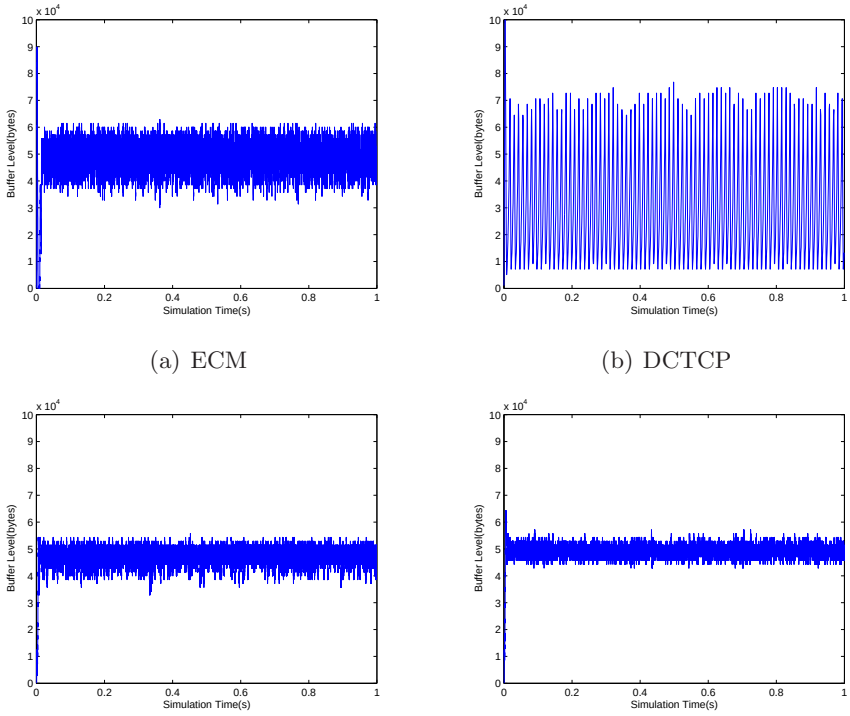
Whereas in the other four cases, with application of respective congestion control schemes, the core switch constantly regulates the buffer utilization through the rate adjustment at the sources, even with excessive generation of data frames from the two sources, the sources manage to hold the frames back from transmitting to the core switch. This traffic regulation successfully avoids frame drop in the core switch, which is indicated by zero frame loss reported in our result. In other words, all the schemes are effective in managing traffic congestion, and our scheme can further provide

Table 3.1: Parameter setting in Experiment 1

Parameters	**Ethernet**	**ECM**	**DCTCP**	**DCC1**	**DCC2**
Q_{eq} / T_{eq}	N/A	50,000B	50,000B	50,000B	50,000B
Q_{mc}	N/A	80,000B	N/A	N/A	N/A
Q_{sc} / T_{sc}	N/A	100,000B	N/A	100,000B	100,000B
Buffer Size	300,000B	300,000B	300,000B	300,000B	300,000B
W, Ru	N/A	W=2 R_u=50Mbps	N/A	N/A	N/A
G_i, G_d	N/A	G_i=0.3 G_d=0.6	N/A	N/A	N/A
AIMD(a_1, b_1) AIMD(a_2, b_2)	N/A	N/A	N/A	AIMD(1, 0.5) AIMD(1, 0.5)	AIMD(1.5, 0.8) AIMD(1, 0.5)
Initial Rate	N/A	1Gbps	N/A	40Mbps	40Mbps
Initial R_{min}	N/A	10Mbps	N/A	N/A	N/A
Initial T_{max}	N/A	1ms	N/A	N/A	N/A
g	N/A	N/A	0.02	N/A	N/A

Table 3.2: Throughput (MB) comparison among five schemes

CM	**Ethernet**	**ECM**	**DCTCP**	**DCC1**	**DCC2**
SR_1	14.9	14.9	14.6	15.0	20.1
SR_2	14.9	14.7	14.6	14.8	9.8
Total Throughput	29.8	29.6	29.2	29.8	29.9
Dropped at CS	22.2	0	0	0	0



(a) ECM (b) DCTCP

(c) DCC1 (d) DCC2

Figure 3.5: Buffer level evolution in the core switch (CS) with various schemes.

differentiated congestion control.

Next we investigate the system stability for the cases with congestion control operations. We focus on the buffer level fluctuation for this study as a fast reaction to a buffer change allows stability of buffer level and avoids potential buffer overflow. From the results given in Fig. 3.5, we see that our scheme has relatively low buffer level fluctuation compared to that of ECM and DCTCP. This result indicates that our scheme has relatively high reaction in controlling traffic congestion. DCTCP records the highest buffer level fluctuation among all, this is mainly because DCTCP sources rely on TCP ACK to indicate congestion where the turnaround time between two end hosts is generally longer. Besides, the new rate adjustment scheme introduced in DCTCP has slower reaction even compared to TCP Reno which also contributes to the high buffer level fluctuation [86].

Table 3.3: Parameter setting in Experiment 2

	Parameters	Value
SR_1	Traffic Generation Rate	240Mbps
	Rate Limiter Setting	AIMD$(1.5, 0.8)$
SR_2	Traffic Generation Rate	240Mbps
	Rate Limiter Setting	AIMD$(1.0, 1.0)$

3.4.2 Experiment 2: Protections with Presence of Misbehaved Hosts

In the second experiment, we consider the situation where LAN sources do not regulate their rates based on the notification instructions given by the CP. They simply ignore the rate cut notifications and keep increasing their transmission rates. This action immediately causes unfairness in the bandwidth sharing at the CP making the congestion control meaningless.

In this experiment, we set the rate limiters of LAN sources such that they do not react to the downward rate adjustment notification. In other words, we use AIMD$(a, 1.0)$ for the LAN sources. Other parameters used in this simulation are listed in Table 3.3. We show the results in Fig. 3.6 and Table. 3.4. Figure 3.6 shows the buffer level in the core switch. Although oscillations exist, the buffer level never goes over T_{sc}. In this simulation, only SAN sources respond to feedback notifications and LAN sources do not cut its rate, this results in over punishment of SR_1 and the ineffective traffic regulation. Instead of reaching a steady state around T_{eq}, the buffer level climbs up even higher due to the excessive LAN traffic. However, at simulation time of 0.33s when the buffer level hits the severe threshold T_{sc}, frame drop mechanism for LAN traffic is triggered to ensure lossless transmission of SAN traffic. We see the buffer level quickly reduces until it reaches the equilibrium threshold T_{eq} again at 0.36s, and the frame drop of LAN traffic is turned off again. From this point, the core switch handles the two sources equally and repeats the cycle of previous operations.

As can be seen from Table 3.4 that the number of frames sent by SR_1 is similar to that of the frames received by DR_1 and the difference between the number of frames sent by SR_2 and that of the frames received by DR_2 matches the dropped frames at the core switch. The results confirm that the mechanism of dropping LAN frames to protect SAN traffic when severe congestion happens is effective.

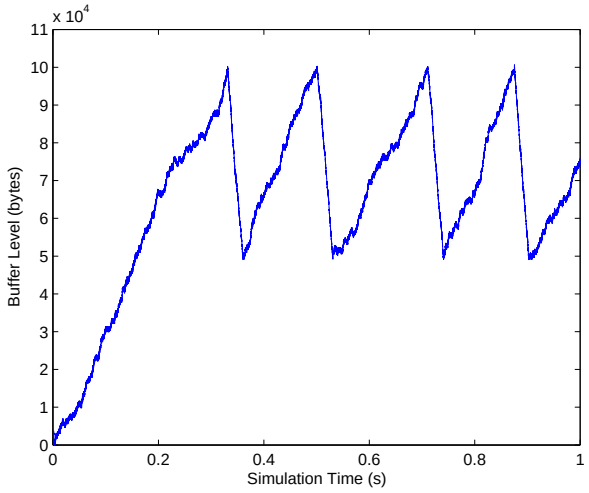


Figure 3.6: Buffer level evolution in the core switch (CS) when excessive traffic arrives to the network from SR_2.

3.4.3 Experiment 3: Bandwidth Utilization Differentiation

In the third experiment, we analyze the bandwidth differentiation feature of our proposed system, using the previous developed analytical model in Section 3.3.2. In addition, the analysis results are compared with simulation results from OMNET++. This study tests the effectiveness of bandwidth differentiation and provides guideline for the implementation of our scheme. We illustrate the bandwidth utilization differentiation by showing the bandwidth differentiation ratio of SAN traffic to that of LAN traffic.

In this study, the AIMD setting for the SAN source is fixed and that of LAN source is varied over a range that is less aggressive than of the SAN source. We report the bandwidth differentiation ratio of SAN traffic over LAN traffic with different parameter

Table 3.4: Number of sent and dropped frames in Experiment 2

Name	**Value** (frames)
Sent from RL_1	5008
Sent from RL_2	4951
Received at DR_1	5008
Received at DR_2	4371
Dropped at CS	580

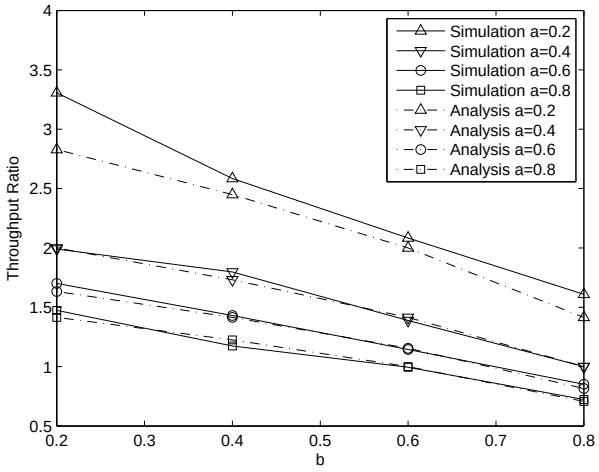


Figure 3.7: Comparison between analytical and simulation results of bandwidth differentiation ratio of SAN traffic with AIMD(1,0.5) to LAN traffic with AIMD(a,b).

settings in Fig. 3.7. As can be seen, varying the parameters a and b of LAN traffic with AIMD(a, b) can provide differentiation in bandwidth utilization. We see that as the values of a and b decrease, the bandwidth differentiation ratio of SAN to LAN traffic increases. We observe that when a switch experiences congestion, it penalizes LAN traffic with AIMD(a, b) where $a < 1, b < 0.5$ more than that of SAN traffic with AIMD(1,0.5). We also observe the potential of AIMD congestion control differentiation where SAN traffic can utilize as high as 3.5 times that of LAN traffic in the network. Under some extreme configurations such as AIMD(0.2,0.2), our analytical results give good agreement to that of the simulation.

To further illustrate the bandwidth differentiation ratio, we show analytical results in Fig. 3.8, where a fixed setting of AIMD(1,0.5) is used for the LAN source while varying AIMD parameter setting for the SAN source. Since SAN traffic has a higher priority, a more aggressive setting than AIMD(1,0.5) for the SAN source is used. Specifically, we use AIMD(a, b) where $a > 1, b > 0.5$. In the following, we demonstrate the design of AIMD settings for a particular desirable ratio with two examples. The first example considers a desired bandwidth differentiation ratio of 2:1, that is if we wish the SAN source to utilize two times higher bandwidth than that of the LAN source. Based on the analytical results given in Fig. 3.8, we may choose AIMD(1.5,0.8) and AIMD(1,0.5) for the SAN and LAN sources respectively. In the second example, we

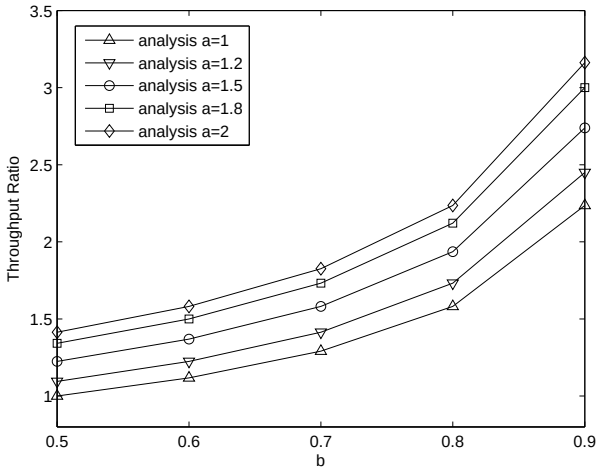


Figure 3.8: Bandwidth differentiation ratio of SAN traffic with AIMD(a,b) to LAN traffic with AIMD(1,0.5) obtained from control theoretic analysis.

Table 3.5: Bandwidth differentiation performance for analysis and simulation results

Case 1	**Case 2**
$a_1 = 1.5, b_1 = 0.8$	$a_1 = 1, b_1 = 0.8$
$a_2 = 1, b_2 = 0.5$	$a_2 = 1, b_2 = 0.5$
Ratio (analysis) = 1 : 1.96	Ratio (analysis) = 1 : 1.58
Ratio (simulation) = 1 : 2.04	Ratio (simulation) = 1 : 1.61

consider a desirable bandwidth differentiation ratio of 1.6:1. According to Fig. 3.8, we may use AIMD(1,0.8) for the SAN source and AIMD(1,0.5) for the LAN source. To confirm that our selections give accurate desirable ratios, we perform simulation measuring the throughput with results showing in Table 3.5. We see that the desirable bandwidth differentiation ratios are indeed achieved.

3.4.4 Experiment 4: Setting of Marking Probabilities

We now study the sensitivity of marking probability on the overall throughput performance at the CP. In our considered system, the core switch marks incoming traffic from different sources with the same probability. As Eqn. (3.5) suggests, varying the value of marking probability does not affect the bandwidth differentiation ratio. Therefore, we

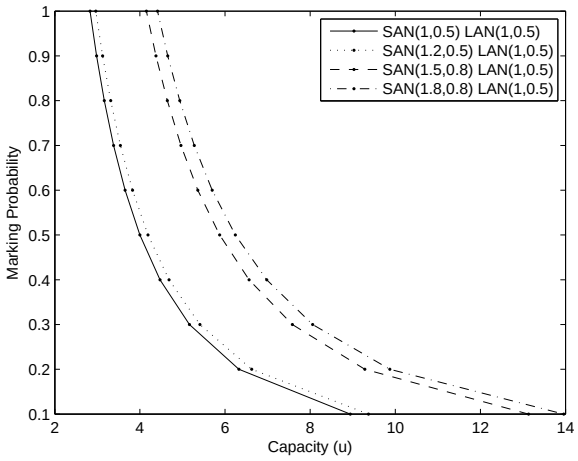


Figure 3.9: The relationship between marking probability and the throughput at the core switch.

can select an optimized probability without compromise on bandwidth differentiation ratio.

As shown in Eqn. (3.6), we have derived the relationship between the marking probabilities and the network capacities. Based on this relationship formula, the best match marking probability can be computed given a certain network capacity. Figure 3.9 shows the capacities of the core switch and their relevant marking probabilities under four different AIMD settings. Specifically, to each given capacity, after acquiring a set of AIMD parameters that satisfies ratio requirements from Fig. 3.8, Fig. 3.9 offers a marking probability demanded by the model accordingly. Although setting a higher marking probability can ensure a stable buffer level around equilibrium threshold as well, this is unnecessary due to the more overhead it may introduce.

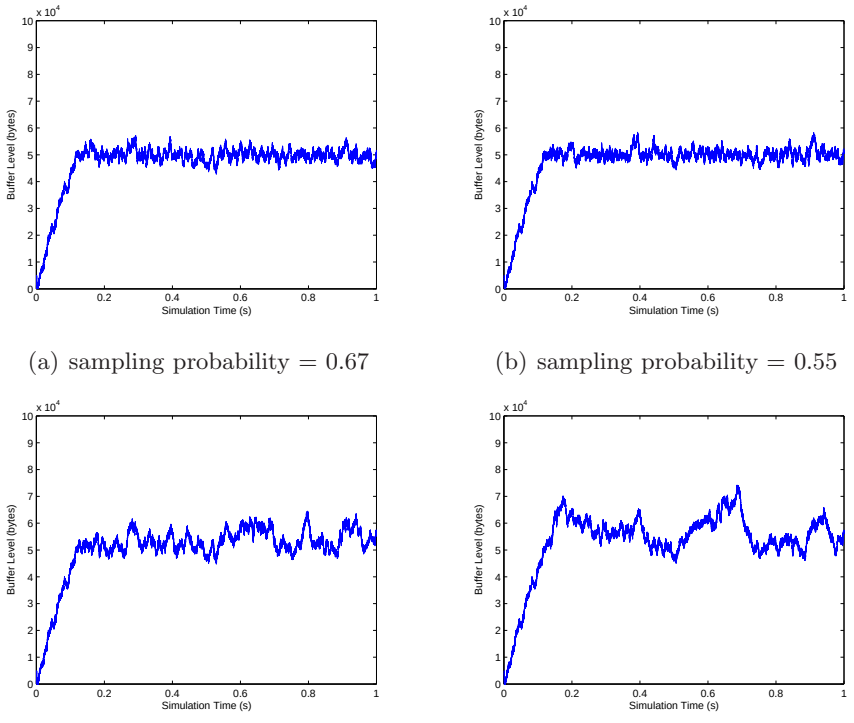
Now with experiment settings given by Table 3.6, we further test the impact on the buffer level with different marking probabilities specified by the term of sampling rates. Similar to the earlier experiments, this scenario considers two SRs sending with a source rate of 240Mbps each, and four STs sending background traffic with a source rate of 4.8Mbps each. We consider a frame length of 1,500 bytes with the processing rate of 20,000 frames per second. Likewise, the processing rate represents the bottleneck of the network capacity where the CS can only deliver 240Mbps of traffic. Excluding the background traffic, the total available network capacity for SRs is 220.8Mbps. With

Table 3.6: Parameter setting in Experiment 4

Parameters	Value
Sampling Rate	2200,2700,3600,4300bytes
SR_1 Rate Limiter	AIMD(1.5,0.8)
SR_2 Rate Limiter	AIMD(1.0,0.5)

$u = 40$Mbps, this gives $C = 5.52$ in the unit of u.

Using our developed formula given in Eqn. (3.6), with $C = 5.52$, an adequate setting of marking probability that maintains high overall link utilization can be determined to be 0.56, and the corresponding sampling rate should be set to 2700 bytes. For performance comparison, we test with the sampling rates of 2200, 2700, 3600, 4300 bytes, which correspond to the sampling probabilities of 0.67, 0.55, 0.42, 0.35, respectively. The simulation results are shown in Fig. 3.10.



(a) sampling probability = 0.67

(b) sampling probability = 0.55

(c) sampling probability = 0.42

(d) sampling probability = 0.35

Figure 3.10: Buffer level in the core switch with different sampling rates.

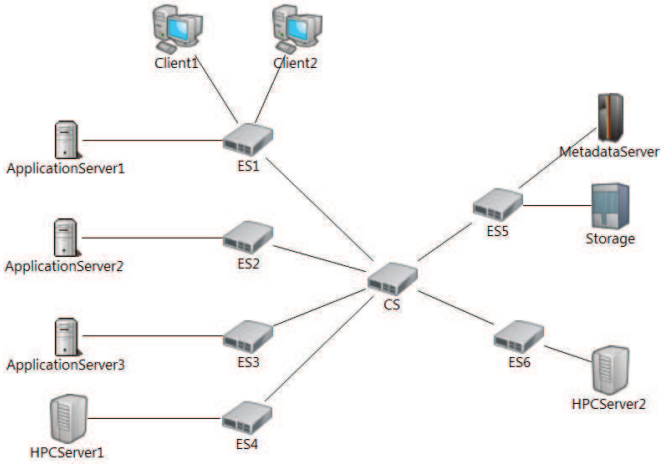


Figure 3.11: Illustration of a case study for multiple sources transmission.

Based on the results presented in Fig. 3.10, it is observed that the marking probabilities of 0.67, 0.55 can maintain a stable buffer level, as shown in Fig. 3.10(a) and Fig. 3.10(b). The comparison between these two figures also shows that sampling more frequently does not significantly improve buffer stability when a stable buffer have achieved. At the same time, increase in sampling rate also results in more overhead in the system. On the other hand, as the sampling probability decreases, the buffer level oscillates in a larger range as shown in Fig. 3.10(c) and Fig. 3.10(d). Though a lower sampling probability incurs less overhead, it leads to system instability and jeopardizes traffic with high priorities. Therefore, sampling rate of 2700 bytes, with corresponding sampling probability of 0.55, is a recommendation setting for the given experiment as we expect.

3.4.5 Experiment 5: A Case Study for Multiple Types of Traffic

Finally, we present a case study for multiple traffic types. The considered network topology is the same as that shown in Fig. 3.4. In this case study, we provide specific roles for each device as shown in Fig. 3.11. Our scenario deals with uploading of files from clients and a total of three types of traffic are involved.

Client1 and Client2 first connect to ApplicationServer1 sending upload requests of LAN traffic type, and this traffic is light. On reception of any request, ApplicationServer1 connects to MetadataServer for data index, and the MetadataServer replies with index to identify and track data storage locations. Traffic load between ApplicationServer1

and MetadataServer can be high, especially with the arrival of multiple simultaneous requests. After index being obtained, Client1 and Client2 connect to the storage directly for file writing, which is SAN traffic. Concurrently, we consider that ApplicationServer2 and ApplicationServer3 also generate index request traffic but for other arbitrary applications, the index requests between the three Application Servers and Metadata Server are considered as LAN traffic. At the same time, HPCServer1 communicates with HPCServer2 with IPC traffic.

In the aforementioned scenario, we select three traffic flows in the purpose of ratio comparison. The first flow comes from ApplicationServer1 to MetadataServer, denoted as *LAN* traffic below. Since TCP takes control of congestion handling, we assign this LAN flow with the lowest priority. Traffic sending from Client1 to Storage acts as the second flow with higher priority, namely *SAN* traffic. Finally, IPC traffic between HPCServer1 and HPCServer2 is the third flow with the highest priority. We adopt a target bandwidth utilization ratio of 1:1.4:1.8 for $\hat{W}_1 : \hat{W}_2 : \hat{W}_3$ where $\hat{W}_1$, $\hat{W}_2$ and $\hat{W}_3$ describe bandwidth utilization for LAN, SAN and IPC traffic.

Following the guideline provided in Experiment 3, we calculate parameters for AIMD and configure them accordingly in the simulations. Thus the targeted ratio can be achieved. Note that there is more than one solution of AIMD parameter settings that can fulfill the targeted ratio, we apply the following method to search for the appropriate AIMD parameter sets. Specifically, we first consider $a_1 = 1, b_1 = 0.5$ for the AIMD setting of the LAN traffic which is also serving as a reference setting to other types of traffic. Given Eqn. (3.5), recursively, we find a particular $a_{i+1} \geq a_i$ and a particular $b_{i+1} \geq b_i$ such that $W_i : W_{i+1} = r_i : r_{i+1}$. We also apply rounding whenever appropriate on the AIMD parameters for the ease of practical operation.

Regarding the simulation settings, the processing capacity is kept the same as the previous setting. For simplicity, we set the main communication parties with average traffic generation rate of 240Mbps each, which includes index requests between three Application Servers and Metadata Server, interprocess computing between the two HPC Servers, and file uploading traffic from clients. Service requests from the clients to ApplicationServer1 are also simulated, but the traffic is comparably small. However, we believe the results of this work are also applicable if they generate traffic at different rates.

Table 3.7: AIMD parameters and achieved ratios for Case Study.

AIMD Parameters	**Achieved Ratios** $(\hat{W}_1 : \hat{W}_2 : \hat{W}_3)$
$a_1 = 1.0, b_1 = 0.5$ $a_2 = 1.2, b_2 = 0.7$ $a_3 = 1.9, b_3 = 0.7$	$1 : 1.46 : 1.78$
$a_1 = 1.0, b_1 = 0.5$ $a_2 = 1.6, b_2 = 0.6$ $a_3 = 1.6, b_3 = 0.75$	$1 : 1.44 : 1.77$
$a_1 = 1.0, b_1 = 0.5$ $a_2 = 1.6, b_2 = 0.6$ $a_3 = 1.9, b_3 = 0.7$	$1 : 1.44 : 1.76$

We illustrate the achieved bandwidth utilization ratios in Table 3.7 for various settings of AIMD parameters. With an ideal mechanism, the targeted ratio of 1:1.4:1.8 should be achieved with high precision. As can be seen from Table 3.7, a certain level of deviations in the ratios is reported for the three settings. This is because our scheme is based on stochastic mechanism where the sampling is random. Nevertheless, the deviations are considered minor for all these settings. This case study involving in three types of traffic again confirms the effectiveness of our design.

3.5 Summary

In view of the demand for a differentiated congestion control for prioritized traffic in DCE, we proposed using Layer-2 end-to-end congestion control for Ethernet congestion management. In this chapter, we introduced the use of different AIMD parameters in rate limiters upon network congestion detected and reported through AQM, and considered the protection of higher priority traffic by detecting and regulating malicious parties. In addition, we presented a comprehensive study confirming the effectiveness of our scheme. Specifically, we employed control theoretic approach to model our scheme with results showing that this method has a potential to allow the SAN traffic utilizing as high as 3.5 times that of LAN traffic throughput. Besides, formulations for recommended parameter settings and sampling probability have been constructed

given network requirements. We also conducted extensive simulation experiments to show the achievement of differentiated congestion control, evaluation with different parameter settings, performance comparison with other schemes as well as a case study in practice.

Chapter 4

Switch Control for Easing Congestion

4.1 Motivation

In Chapter 2, we discuss that switch control can easily acquire network traffic status and make dynamic routing decisions. In this chapter, we propose a switch control scheme to ease congestion with L2MP capacity. As the communication infrastructure in a data center is typically a hierarchical tree, multiple redundant physical links are deployed between core/aggregation and aggregation/access layers. The redundancy not only prevents single point of failure (SPOF) [20], [21] but also increases network capacity. However, the network capacity cannot be fully utilized by STP used in conventional Ethernet, STP blocks some links and ports to avoid forwarding loops. To increase the bisectional bandwidth [1] between communication for pairs, replacing schemes become inevitably necessary for DCE. Some research works have proposed solutions to address this issue, details can be referred in Chapter 2.

L2MP, as a viable replacement of STP, can alleviate congestions by increasing the utilization of network redundancies. Unlike STP, L2MP does not block any link and port from active usage. It makes use of the multipath capability of the underlying

[1] Bisectional bandwidth defines the bandwidth between the two segmented equal parts in the network. Typically, this refers to the worst-case segmentation, but being of equal parts is critical to the definition, as it refers to an actual bisection of the network. The bisection bandwidth of a network minimizes such bandwidth along all possible bisections. Therefore, bisection bandwidth can be used as an effective way in measuring worst-case network capacity.

network topology by spreading traffic between two communication pairs along multiple paths [20], [21].

To address this issue, we propose a switch control L2MP mechanism, Dynamic Load Balancing MultiPathing (DLBMP), to achieve multipath capability in DCE. With such capability, network congestions can be eased by dynamically distributing traffic. This scheme consists three algorithms, namely *multipath construction algorithm*, *path load updating algorithm* and *multipath routing algorithm*. The multipath construction algorithm calculates multiple routes and initialize them at access switches for each source-destination pair. The path load updating algorithm accumulates path load information and updates it to access switches. With the path load information collected, the multipath routing algorithm selects optimal route for traffic delivery.

Admittedly, DLBMP can significantly improve bisectional bandwidth. However, it seems helpless in preventing traffic loss eventually if a network is overloaded. We expect that either DLBMP or congestion control (CC) alone is inadequate in traffic handling in terms of both lossless delivery and high throughput. One case is that congestion may occur within the network due to routing conflicts. In other words, multiple flows destined to different end stations are routed through the same switches, resulting in overutilization of partial links in the network and underutilization of the rest. In such a case, reducing the source rate by activating the corresponding CC mechanism can resolve the congestion but result in performance degradation. A viable solution is to utilize the multipath capability of DCE and divert some traffic flows to a light-loaded path. On the other hand, if the congestion comes from contention of end stations or extremely bursty traffic, reducing source rate presents a better solution. This might happen when multiple sources are sending to the same destination with heavy load that exceeds the link capacity, or all viable routes are fully occupied by the heavy bursty traffic load. Therefore, DLBMP achieves even and optimal network resource utilization, while CC prevents excessive traffic from being pumped into the network.

A well-designed dynamic combination of DLBMP and CC theoretically should perform better than each individual scheme, since the combined considers both key factors that affect network performance. However, a simple merger of two schemes might yield poor performance due to duplication of overhead and system delay. Firstly, this is because DLBMP normally balances its load based on link utilization as an indication,

it appears to be different from CC which takes buffer level as an indication. The duplication of overhead further adds more pressure to the overloaded network. Moreover, a lag exists between indication and reaction towards congestions, thus the congestion may not be eased timely. Both overhead and delay increase the system complexity and result in a less effective performance.

The above observations motivate us to propose a switch control scheme by integrating DLBMP with CC mechanism, namely **DLBMP with CC**. DLBMP accommodates traffic flows on multiple paths, while CC avoids admission of excessive traffic. This combined scheme, ensures fair, efficient and optimal network utilization, and hence ensures the performance in terms of both lossless frame delivery and network throughput. Instead of using two separate parameters, i.e. link load and buffer level, to trigger DLBMP and CC, our switch control scheme only needs to monitor the link load and can manage DLBMP and CC in a more integrated and efficient way. We further decouple this control overhead into a distinct central controller by SDN technology. In addition, our switch control scheme introduces application-layer flow differentiation. With such a fine flow differentiation (FFD) mechanism, traffic can be more evenly distributed among multiple paths, resulting in better bandwidth utilization. Simulation results show that our combined scheme can further improve network throughput with the FFD mechanism and guarantee lossless delivery with the integrated CC.

The rest of this chapter is organized as follows. Section 4.2 describes the major mechanisms of our switch control scheme, DLBMP with CC. Section 4.3 presents computer simulation results of our proposed scheme and comparison with other schemes. Finally, a brief summary concludes this chapter in Section 4.4. This research work has been summarized and published in [87], [88].

4.2 An Integrated Two-tier System Design

4.2.1 Term Definition

Here we firstly present frequently used terms in this part.

- DLBMP cloud: DLBMP cloud refers to a group of connected switches running DLBMP protocol in a data center Ethernet infrastructure.

- Access switch: access switches refers to switches having direct connection with end stations or with switches running other forwarding protocols, as illustrated in Fig. 2.2.

- Non-access switch: accordingly, non-access switches are those switches which do not have direct connection with end stations or switches running other forwarding protocols. In a typical DCE topology, non-access switches include both core layer switches and aggregation layer switches.

- Ingress switch and egress switch: each data frame traveling through the DLBMP cloud has its own ingress switch and egress switch. An ingress switch refers to the switch from which the data frame enters the DLBMP cloud. An egress switch refers to the corresponding switch from which the data frame leaves the DLBMP cloud.

4.2.2 Route Construction Algorithm

4.2.2.1 Initial Path Computation

DLBMP switches run a link state protocol among themselves. A link state protocol is one in which connectivity is broadcast to all switches, so that each DLBMP switch knows about all the other switches and the connectivity between them. Then DLBMP utilizes such information to compute pairwise optimal paths to every other switch. DLBMP maintains and utilizes all discovered shortest paths, i.e., the equal cost multi-paths (ECMP). In DLBMP, hop count is used as the cost metrics for the initial route computation because in a 10GE data center usually all links are with the transmission speed of 10Gbps. Theoretically, traffic flows can also be delivered through non-equal cost paths (longer paths), however this is not considered in DLBMP at the current stage since such a method provides marginal advantages in a typical DCE topology but increases the routing complexity significantly [26].

4.2.2.2 Route ID Generation and Reconcilement

In the initial stage, each switch needs to be pre-configured as an access switch or a non-access switch, and such attribute for a switch is recorded in its database accordingly.

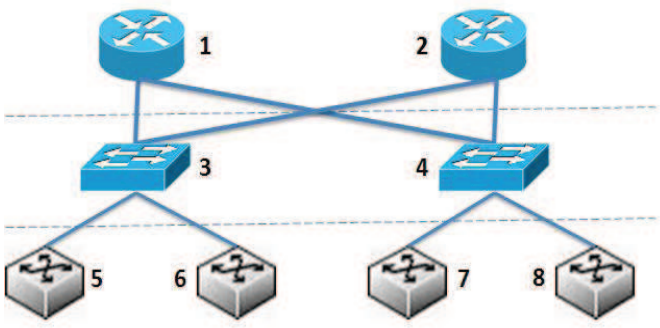


Figure 4.1: Data Center interconnection topology example.

After an access switch finishes its route computation, it generates a unique *routeID* with the form of *srcMAC.destMAC.RouteNo* for each discovered route. For example, corresponding to the topology in Fig. 4.1, the access switch S_5 has two routes to the access switch S_7, 5-3-1-4-7 and 5-3-2-4-7 respectively with *routeID* of $MAC_5 \cdot MAC_7 \cdot 01$ and $MAC_5 \cdot MAC_7 \cdot 02$ accordingly. For the ECMP group for a specific access switch-access switch pair, *RouteNo* always starts from 1 for the first ECMP route and is added by 1 for each next route. An access switch keeps the generated *routeID* for each discovered route in its *routingTable*, and in the initial route construction phase a *routingTable* includes three information fields for each entry, destination, route *routeID* respectively. During the operational phase, more fields will be added to the *routingTable*, which will be explained in detail in Section 4.2.3.1. Table 4.1 shows parts of the initial *routingTable* of S_5 corresponding to Fig. 4.1. Each access switch floods the network with its *routingTable* to help reconcile the *routeID* information at other switches.

A non-access switch also maintains its *routingTable* after the route computation,

Table 4.1: *routingTable* of S_5 in the initial phase

Destination	**Route**	***routeID***
MAC_7	3–1–4–7	$MAC_5 \cdot MAC_7 \cdot 01$
MAC_7	3–2–4–7	$MAC_5 \cdot MAC_7 \cdot 02$
MAC_8	3–1–4–8	$MAC_5 \cdot MAC_8 \cdot 01$
MAC_8	3–2–4–8	$MAC_5 \cdot MAC_8 \cdot 02$
...	...	...

Table 4.2: Initial *routingTable* of S_3 before updating

Destination	**Route**	***RouteID***
MAC_7	1–4–7	-
MAC_7	2–4–7	-
...	...	...

however, it itself does not generate *routeID* for discovered routes but updates the *routeID* entry for each recorded route in its *routingTable* according to the ones received from access switches.

How a non-access switch proceeds upon receiving the *routingTable* from an access switch is elaborated with the following example: Table 4.2 presents partial *routingTable* of S_3 before it receives *routingTable* from any access switch. Supposing S_3 then receives *routingTable* from S_5, it compares the two tables entry-by-entry. Since the first entry in both tables, namely S_3_*entry1* and S_5_*entry1*, gets the same destination field, MAC_7, the route field of the two entries is then compared. When S_3 finds out that the latter part of the route field of S_5_*entry1* (the part after its own ID in the route field, specifically '1-4-7' here) identically matches with the route field of S_3_*entry1*, it copies the *routeID* (i.e. $MAC_5 \cdot MAC_7 \cdot 01$) of S_5_*entry1* to the *routeID* field of S_3_*entry1*. After S_3 updates table entries by receiving *routingTable* information from S_6, one more *routeID* will be added to the *routeID* field of S_3_*entry1*, indicating the route from S_5 to S_7 and the one from S_6 to S_7 share a common part between S3 to S7. Table 4.3 illustrates the *routingTable* of S_3 after updating.

The purpose of maintaining such consistent *routeID* information among all switches is that a switch can uniquely map a *routeID* to a route recorded in its *routingTable*. When a data frame enters the DLBMP cloud, the access switch chooses one route to

Table 4.3: *routingTable* of S_3 after updating

Destination	**Route**	***RouteID***
MAC_7	1–4–7	$MAC_5 \cdot MAC_7 \cdot 01$; $MAC_6 \cdot MAC_7 \cdot 01$
MAC_7	2–4–7	$MAC_5 \cdot MAC_7 \cdot 02$; $MAC_6 \cdot MAC_7 \cdot 02$
...	...	...

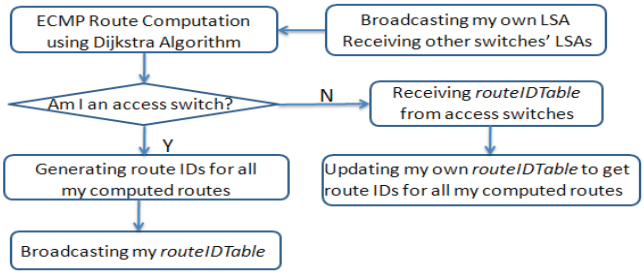


Figure 4.2: Flowchart of DLBMP initialization procedure.

forward the frame and attaches the *routeID* of the chosen route in the frame header. As an intermediate switch receives the frame, it simply forwards the frame to its next hop according to the *routeID* by consulting its *routingTable*. When the corresponding access switch gets the frame, it checks its MAC association table and forwards the frame to the destined end station through the correct Ethernet port.

Figure 4.2 uses a flowchart to summarize the initial route computation and *routingTable* construction procedures of DLBMP.

4.2.3 Path Load Updating Algorithm

In our design, path loads of routes are updated through two phases. In the first phase, a data frame piggybacks the highest link load along intermediate switches on the route to its egress switch. In the second phase, ingress switches receive path load updates through central controller which periodically exchanges control information among access switches.

4.2.3.1 Upstream Update Using Piggybacking

During operation, each switch continuously measures the number of outgoing frames and the size of each frame from its Ethernet ports, so that it can determine the occupied link capacity for all attached links. For a link L, the ratio between utilized link capacity and total link capacity is defined as the link load l_L for L. Hence, for a path P composing of several links $(L_1, L_2, ..., L_n)$, the corresponding path load l_P is defined as follows:

$$l_P = max(l_{L_1}, l_{L_2}, ..., l_{L_n}) \tag{4.1}$$

where $l_{L_i}(i = 1, 2, ..., n)$ represents the link load of $L_i(i = 1, 2, ..., n)$ respectively.

As introduced before, it is the assignment of an ingress switch to choose the route for delivering a data frame received from some end station. When the forwarding route is selected, the ingress switch piggybacks some extra information, namely *pathLoad*, in the frame header. Here, *pathLoad* refers to the transient link load of the corresponding outgoing link, which forms the first hop on the selected route (Step 2 in Fig. 4.3).

When a non-access switch along the selected route receives the corresponding frame, it compares the recorded *pathLoad* in the frame header with the link load of its outgoing link, which is the next hop for forwarding the frame. If its link load is lower than *pathLoad*, it keeps the *pathLoad* field of the data frame intact; otherwise it replaced *pathLoad* with its own link load. Then the intermediate switch forwards the frame to its next hop based on *routeID* information in the frame header. With such means, *pathLoad* in the frame header can always record the *linkLoad* of the most heavily loaded link along the path (Step 3 in Fig. 4.3). As the corresponding egress switch receives the data frame, it can obtain the corresponding *pathLoad*. Then periodically the egress switch can update its ingress switches about the *pathLoad* information of their associated routes by sending explicit updating messages to the corresponding ingress switches (Step 4 in Fig. 4.3).

Here we give an example by using the network topology of Fig. 4.1 to demonstrate how the path load is updated in DLBMP: supposing S_5 is sending a data frame to S_7 (actually this data frame is originated from some end station attached to S_5 and is supposed to deliver to some end station attached to S_7) and route 5-3-2-4-7 is selected. First S_5 attaches the corresponding *routeID*, $MAC_5 \cdot MAC_7 \cdot 02$, in the frame header, fills the *pathLoad* field with the link load of its outgoing link 5-3, and delivers the frame to its next hop S_3. As S_3 receives the frame, it compares the transient link load of its outgoing link 3-2 with the *pathLoad* information indicated in the frame header. If the link load of link 3-2 is higher, it replaces the *pathLoad* information with the corresponding link load before forwarding the frame to its next hop S_2, otherwise it simply delivers the data frame to S_2. S_2 and S_4 follow the same procedure until the data frame reaches the destination switch S_7. Consequently S_7 obtains the *pathLoad* information of route $MAC_5 \cdot MAC_7 \cdot 02$. In the meanwhile, S_7 periodically updates S_5 with the *pathLoad* information for all routes originated from S_5 and ended at S_7.

An example *routingTable* of S5 corresponding to Fig. 4.1 is presented in Table 4.4.

Table 4.4: *routingTable* of S_5

Destination	**Route**	***routeID***	***pathLoad***	**rtBound**	**lastUpdate** (s)
MAC_6	3–6	$MAC_5{\cdot}MAC_6{\cdot}01$	40%	[0.00, 1.00)	101.5
MAC_7	3–1–4–7	$MAC_5{\cdot}MAC_7{\cdot}01$	80%	[0.00, 0.33)	102.0
MAC_7	3–2–4–7	$MAC_5{\cdot}MAC_7{\cdot}02$	40%	[0.33, 1.00)	102.0
...	...	...	...	...	...

The *lastUpdate* field records the time of receiving the last updating message for the corresponding entry. The *rtBound* field decides the probability of using the corresponding route, and it is related with the *pathLoad* field. And their relationship is given as follows:

$$p_{P_j} = \frac{1 - l_{P_j}}{\sum_{k=1}^{M} 1 - l_{P_j}} \tag{4.2}$$

$$\begin{cases} RB_{P_j} \in \left[0.00, p_{P_j}\right), & j = 1 \\ RB_{P_j} \in \left[\sum_{i=1}^{j-1} p_{P_i}, \sum_{i=1}^{j} p_{P_i}\right), & j \geq 1 \end{cases}, \tag{4.3}$$

where M indicates that a total of M routes are available for the destination, (P_1, P_2, ..., P_j, ..., P_M) defines the route set, and P_j can represent any route in the route set. Also, the route set is ordered according to the *routeID* of each route in the set. The quantity of p_{P_j} is the probability of using route P_j in the presence of all M routes with individual path load, and RB_{P_j} gives *rtBound* for P_j. Since the *pathLoad* field is updated once associated updating message is received, the corresponding *rtBound* field will be updated accordingly.

For example, the first entry in Table 4.4 records a route to the switch S_6. S_5 always uses it to forward data frames to S_6 as it is the only shortest path to S_6. The second and third entry maintains two different routes to S_7. Based on the transient *pathLoad* of the two routes and the calculation of Eqn. (4.2), the probabilities of utilizing $MAC_5{\cdot}MAC_7{\cdot}01$ and $MAC_5{\cdot}MAC_7{\cdot}02$ are 33.3% and 66.7% respectively.

4.2.3.2 Downstream Update Using SDN

Access switches connect to a central controller which enables *pathLoad* exchange among access switches without network delay. Instead of running a sole distribution algorithm,

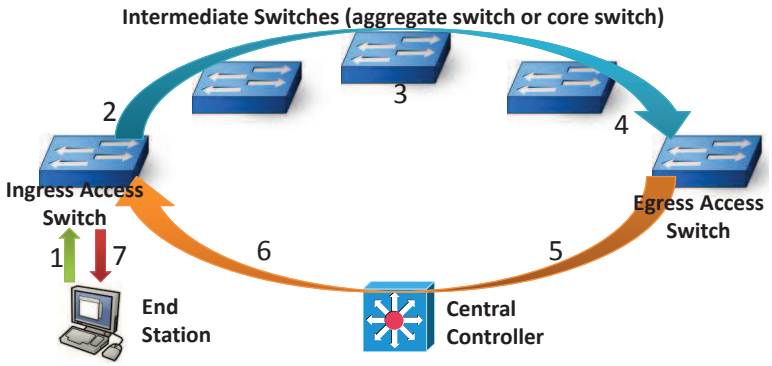


Figure 4.3: Illustrations of path load updating.

which mixes data traffic and control information with each other and results in performance degradation, or a sole SDN protocol, which depends only on a central controller that accesses to every switch and controls all the intelligence, our design lets a data frame piggyback control information in its frame header in the data path on the way to its egress switch, while ingress switches obtain *pathLoad* updates through the control path.

Specifically, only access switches in our proposal runs SDN protocols, which allows the central controller to access *routingTable*s of all the access switches. When SDN-enabled switches start, they open a secure channel to the central controller. The controller can query, insert and modify flow entries. The switches maintain statistics in *routingTable*s, such as *pathLoad* and *lastUpdate* as illustrated in Table 4.4.

The central controller performs the following tasks where it obtains network status from *routingTable*s in all access switches periodically, and updates ingress switches accordingly when it detects *pathLoad* change on the associated routes (Step 5 and step 6 in Fig. 4.3).

The switches perform the following tasks where if an incoming frame does not match any of the flow entries in the *routingTable*, the switch inserts a new flow entry with the appropriate output port (based on Section 4.2.4.3) which allows any subsequent frames to be directly forwarded at line rate in hardware (Step 1 in Fig. 4.3). Once traffic volume on a route grows beyond the specified threshold T, the access switch may inform the source to adjust its data rate according to the mechanism discussed in Section 4.2.5.

To summarize, an illustration of updating process is presented in Fig. 4.3. The process follows seven steps, including Step 1, ingress switch frame forwarding; Step 2, ingress access switch route selection; Step 3, intermediate switch path load piggybacking; Step 4, egress switch path load updating; Step 5, central controller path load updating; Step 6, ingress switch path load updating; and optionally Step 7, CC feedback if path load exceeds preset threshold.

Our design differentiates itself with a central algorithm in two ways. First, only access switches are SDN-enabled and the rest can be common switches. This setup permits easy implementation and high scalability. Second, it also runs distributed algorithm in case of controller failure, where updates of ingress switches are done through data paths, this ensures that our system operates even when the controller cannot function well.

4.2.4 Traffic Routing Algorithm

4.2.4.1 Hieratical Forwarding

When a frame enters the DLBMP cloud, the handling ingress switch only needs to know how to route it to the corresponding egress switch instead of the destination station. Hence, a DLBMP switch only needs to compute routes to other switches but not to end stations.

The MAC addresses of end stations only need to be associated with the addresses of the switches to which the end stations are attached, and the association is only made aware to those access switches. An ingress switch consults the central controller to acquire the associated egress switch address for a certain end station. This association is kept in the ingress switch cache for further reference. Hence, as a frame enters the DLBMP cloud, it is routed based on the addresses of ingress/egress switch pair instead of the addresses of end stations, and this is called hierarchical forwarding [26]. Hierarchical forwarding utilized in DLBMP allows switches, which do not have direct connections to end stations, to maintain smaller forwarding tables, and thus it facilitates DLBMP to scale better.

To be routed within a DLBMP cloud, a frame needs to carry some additional information, such as MAC addresses of the ingress and egress switches. Such information is piggybacked as an extra header to the original Ethernet frame so that the original

frame generated by an end station remains intact. Fig. 4.4 shows the format of a frame with the extra header within a DLBMP cloud. The carried extra fields, *routeID* and *pathLoad* specifically, are introduced in details in Section 4.2.2 and Section 4.2.3.

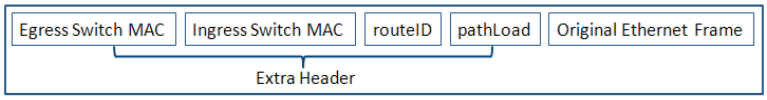


Figure 4.4: Illustration of DLBMP frame format.

4.2.4.2 Per-flow Forwarding

DLBMP provides dynamic load balancing by splitting traffic across multiple paths. However, if paths have different delays, split traffic at packet granularity can cause a large number of frames arriving out-of-order. TCP confuses this reordering as a sign of congestion, resulting in degraded performance [89], [90]. Even some UDP-based applications are sensitive to packet reordering. Most importantly, it is critical to preserve the in-order delivery for Fibre Channel over Ethernet (FCoE) SAN traffic as neither FC nor Small Computer System Interface (SCSI) can handle packet reordering well. In DLBMP, all frames of a flow are forwarded on the same path to preserve in-order delivery and traffic splitting for load balancing purpose is only occurred at flow granularity.

The typical frame format for an Ethernet frame is shown in Fig. 4.5, and it can be observed that the MAC header includes three information fields, destination MAC address, source MAC address and EtherType. The EtherType field identifies the payload type: a value of 0x0800 indicates the corresponding frame carries an IPv4 packet, and a value of 0x8906 indicates an FCoE packet is piggybacked.

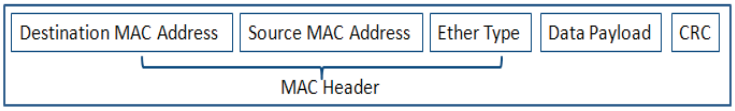


Figure 4.5: Illustration of typical Ethernet MAC frame.

Beside using MAC address and Ethertype in Ethernet frame header, application-layer related information is also considered in flow differentiation such that flows can be

differentiated with finer granularity. For example, IPv4 packets can be further differentiated by its TCP or UDP port number, and FCoE flows can be further differentiated by their OXID along with other SCSI exchange parameters. As a result, application-layer flows can be finely differentiated in DLBMP.

In DLBMP, each access switch records all the flow that it currently handles in an information table named *flowInfoTable*. Table 4.5 gives an example *flowInfoTable*. *flowID* field is utilized to uniquely identify an application-layer flow, and actually it represents the header portion of all frames, which belongs to this flow. The *routeID* field gives the route used for the corresponding flow, and the *lastUpdate* field records the time of the last frame received for the flow to help validate the freshness of a flow. Non-access layer switches do not maintain such *flowInfoTable* as they always forward a frame according to the *routeID* indicated in the header of the received frame.

As a data frame enters the DLBMP cloud through some access switch, the access switch can tell whether the frame belongs to an existing flow or not. If the frame belongs to an existing flow, the access switch first updates the *lastUpdate* field with current time of the entry corresponding to the flow in the *flowInfoTable*, and then uses the route referred by the *routeID* field to deliver the current data frame. In addition, it attaches the corresponding *routeID* in the frame header. Otherwise, it selects a route to forward the frame and adds an entry for the new flow in its *flowInfoTable* as the frame is likely the first frame for a new flow.

4.2.4.3 Load Balancing

We know that it is always the assignment of an ingress switch to decide how to route a data frame, which enters the DLBMP cloud from it. As long as an ingress switch receives a frame from some end station, it first checks whether the frame belongs to

Table 4.5: FlowInfoTable of S5

FlowID	***RouteID***	**LastUpdate**
MAC_{host1}-MAC_{host3}-*IPv4-TCP-portA*	$MAC_5 \cdot MAC_7 \cdot 01$	99.6s
MAC_{host1}-MAC_{host3}-*FCoE-OXID1*	$MAC_5 \cdot MAC_7 \cdot 02$	99.2s
MAC_{host2}-MAC_{host3}-*IPv4-UDP-portB*	$MAC_6 \cdot MAC_7 \cdot 02$	99.5s
...	...	...

an existing data flow. If true, it uses the same route, which has been utilized for previous data frames of the flow. Otherwise, DLBMP algorithm applies a random number generator (RNG) to obtain a number n, where $n \in [0.00, 1.00)$. Based on the value of n, the corresponding route for the new flow can be decided. For example, as shown in Fig. 4.1, when S_5 receives a data frame destined to some end station associated with S_7 and discovers that the frame does not belong to any existing flow, it then selects a route for the new flow by utilizing its RNG. Based on Table 4.4, there are two available routes for S_7 and their corresponding *rtBound* are $[0.00, 0.33)$ and $[0.33, 1.00)$ respectively. Supposing the generated $n = 0.82$, the route selected for the new flow will be $MAC_5 \cdot MAC_7 \cdot 02$. Accordingly, S_5 adds an entry for the new flow in its *flowInfoTable*.

Such a mechanism promises that when a new traffic flow injected into the DLBMP cloud, it is more likely forwarded through a more lightly-loaded route. Hence, traffic load between a communication pair can be more evenly split on all the available routes, resulting in fairer link utilization and more load-balanced routing, while preserving per-flow forwarding property.

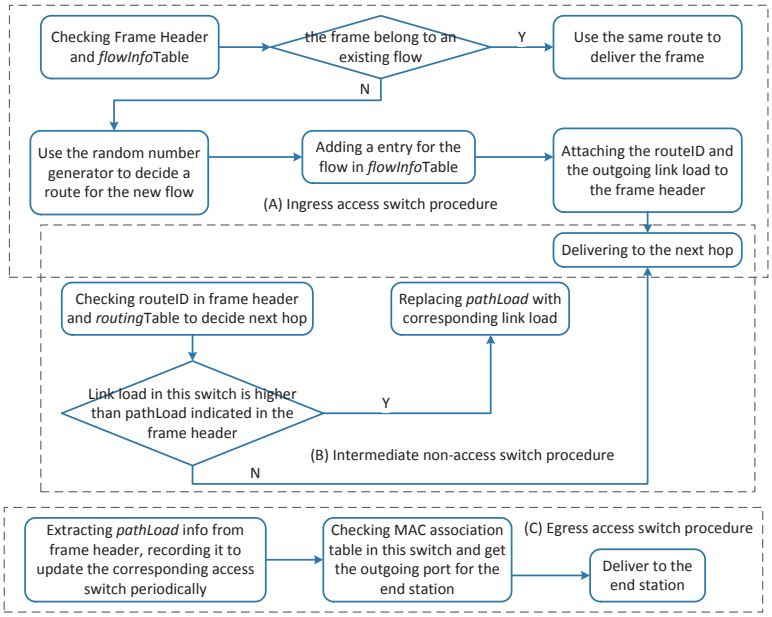


Figure 4.6: Flowchart of data frame handling in DLBMP.

Based on the scheme description from Section 4.2.4.2 to 4.2.4.3, we are able to understand how a data frame is handled when it is within the DLBMP routing cloud.

Fig. 4.6 uses flowcharts to demonstrate how switches with different attributes handle a data frame, i.e., respectively, how an ingress switch, an intermediate switch and an egress switch handles a data frame.

4.2.5 DLBMP with Source Congestion Control

By integrating the source congestion control scheme into DLBMP, we aim to better handle network congestion and frame dropping by also controlling data rate from data sources. Details of design and implementation of source congestion control can be referred to Chapter 3. Here we highlight some important mechanisms.

4.2.5.1 Congestion Detection and Notification

Unlike the mechanism in Chapter 3, in which congestion notification is generated from any switch with potential congestions, here the access switches are equipped with intelligence for detecting congestion on a route and notifying corresponding data sources with explicit messages. That is, in DLBMP, each access switch records up-to-date *pathLoad* for each route originated from itself, thus it can easily detect a congested route when *pathLoad* keeps increasing and exceeds a predefined threshold T. Then the access switch informs all the data sources on that route according to its *flowTable*, requiring the data sources to reduce the rates of their flows.

We implement a sample unit for a certain route, which controls notification intervals. In case of a potential congestion upon receiving an update exceeding T, it starts recording the number of bytes that has been sent through this route. When a frame arrives and the accumulated bytes reach a preset value S bytes, sample unit selects this frame and informs the source for rate reduction. Using this technique, our scheme maintains a certain percentage of marking rate, and a source with higher data rate stands a higher probability for rate cut.

4.2.5.2 Rate Limiter Control

At source side, we define generation rate as real traffic generating rate from application layer, however, to avoid excessive flow entering into network, a rate limiter is implement at sources, which controls network traffic load with a regulated sending rate.

Initially, a rate limiter controls its sending rate by a and increases it exponentially by doubling the rate in each predefined time interval called a *slot*. Upon receiving congestion feedback, the rate limiter enters into an adjustment cycle. Similar to AIMD in TCP, we implement our rate control in such a way that once a source receives a congestion feedback it will have its rate cut to a fraction b of its previous rate. After that, the source increases its rate linearly by a in every slot.

Let $x_i(t)$ be the sending rate flow i transmits to the DLBMP cloud at time t through a certain route, source i will adjust its load at the next slot by

$$x_i\left(t+1\right) = \begin{cases} a_i + x_i\left(t\right), & X\left(t\right) \leq T \\ b_i x_i\left(t\right), & \text{otherwise} \end{cases} \tag{4.4}$$

where $X(t) = \sum_i x_i(t)$ is the sum of traffic loads transmitting through the same switch, T is a preset threshold indicating a potential congestion. Note that AIMD can be described by two parameters, a and b respectively where $a > 0, 0 \leq b < 1$.

4.3 Performance Evaluation and Discussion

In this part, we use a network simulator OMNET++ 4.1 [84] to evaluate the performance of our proposal by using an interconnected topology as shown in Fig. 4.7. We conduct two experiments to test the effect of important system parameters, such as reduction ratio b and threshold T, to offer parameter guidelines. One experiment focuses on a single flow level of the impact on throughput with ratio b and threshold T. The other experiment focuses on multiple flows of the impacts on threshold T with L2MP routing. After that, we further conduct two experiments to compare performance gaps on four respective schemes including STP, TRILL, and our single DLBMP mechanism, denoted as DLBMP, as well as, the integrated switch control scheme with both FFD and CC, denoted as DLBMP+FFD+CC. One experiment compares frame delivery ratios of the four schemes. The other experiment compares their load balance capabilities.

Fig. 4.7 shows a typical data center network topology used for the computer simulations. SW1, SW2, SW9 and SW10 are classified as access switches to which end stations are directly connected. Each access switch and its connected end station group form a Pod, which means Pod1 consists of SW1 and Host1-Host4, Pod2 consists of SW2

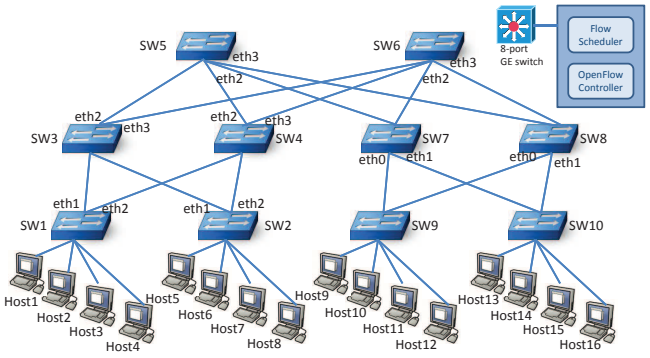


Figure 4.7: Network topology of the simulation testbed.

and Host5-Host8, Pod3 consists of SW9 and Host9-Host12, and Pod4 consists of SW10 and Host13-Host16. The basic simulation parameter settings are listed in Table 4.6.

Table 4.6: Basic parameter setting.

Name	Value
Link Rate	10Gbps
Sample Byte S	20KB
Notification Probability P_r	0.5

4.3.1 Experiment 1: Single Flow System Parameter Test

In the first experiment, we test the impact of reduction ratio b and threshold T on the network throughput.

We run a simple simulation scenario with only one flow from Host1 to Host9, starting at frame interval of 1ms at 10s, which means the time interval between two messages is 1ms. The interval reduces 5 times every 10 seconds, that means, it changes to 0.2ms on 20s, 0.04ms on 30s and so forth. In other words, the flow volume increases on every 10 seconds. The simulation runs for 60 seconds. In this test, there are eight simulation runs with the parameter sets of $b \in \{0.5, 0.6, 0.7, 0.8\}$ and $T \in \{8, 9\}$. Then we compare their throughput difference according to the results we obtain.

Details of the simulation parameters are listed in Table 4.7 and we report network throughput in Fig. 4.8. As can be seen, keeping the parameter b unchanged, an increase of threshold T results in a higher throughput. Similarly, an increase of b while

Table 4.7: Parameter setting in Experiment 1

Name	Value
Simulation Time	60s
Threshold T	[8, 9]Gbps
Cut Ratio b	[0.5,0.6,0.7,0.8]
Flow Interval	1ms,0.2ms,0.04ms,0.008ms,0.0016ms

maintaining the same T also yields an increase of throughput.

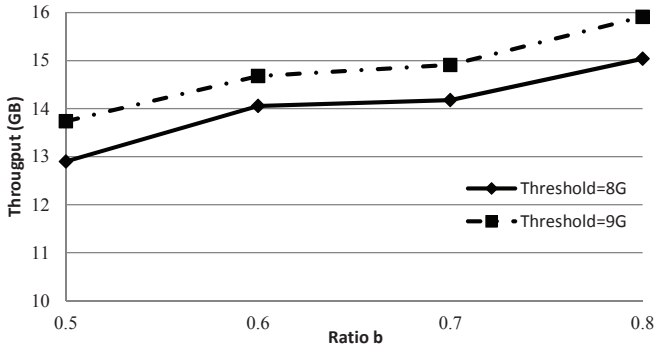


Figure 4.8: Throughput of the single flow with regard to T and b.

4.3.2 Experiment 2: Multiple Flows System Parameter Test

In the second experiment, we evaluate the impact of T on system throughput with multiple flows transmitting from Host1-Host8 to Host9-Host16. These flows randomly start from 10s to 15s and end randomly before 100s with interval randomly set from 0.1ms to 1ms. In the simulation run of 100s, we set two scenarios with different flow numbers of 1700 flows and 2500 flows. Each consists simulation runs with threshold 7Gbps, 8Gbps, 9Gbps and 10Gbps. Note that 10Gbps means no control at sources. Simulation parameters are shown in Table 4.8.

Throughput of different parameter settings is reported in Fig. 4.9. We can see from the figure that as threshold increases, higher throughput is yielded for both 1700 flows and 2500 flows. Although higher throughput can be obtained with a higher threshold, it stands a risk of frame drop due to the delay control towards congestion. As we observe, there is a slightly drop of 0.06% in the simulation run of 2500 flows with threshold of

Table 4.8: Parameter setting in Experiment 2

Name	Value
Simulation Time	100s
Flow Number	[1700, 2500]
Threshold T	[7, 8, 9, 10]Gbps
Cut Ratio b	0.5
Sources	Host1-Host8
Destinations	Host9-Host16
Flow Type	0-199
Start Time	10-15s
End Time	10-100s
Flow Interval	0.1-1ms

9GBps. In the next two experiments, we choose 8Gbps as threshold for the next test to obtain high throughput with no frame drop.

4.3.3 Experiment 3: Frame Delivery Ratio Comparisons

In the third experiment, we compare performance of respective schemes in terms of frame delivery ratio of STP, TRILL, DLBMP, and DLBMP+FFD+CC with different numbers of flows.

Using the suggested parameters from Experiment 1, we run simulations in which 1700, 1900, 2100, 2300 and 2500 flows are generated from Host1-Host8 sending to

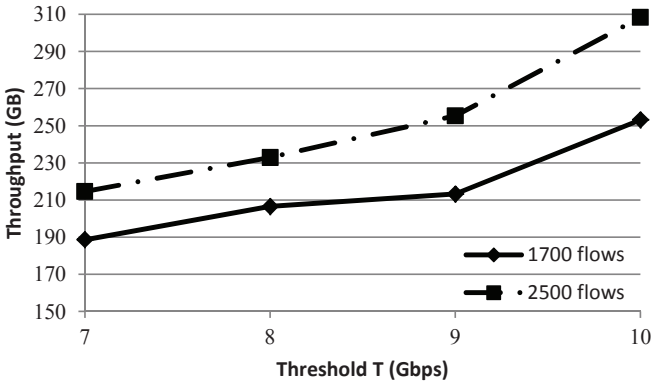


Figure 4.9: Throughput with regard to T on multiple flows.

Host9-Host16. In this experiment, the simulation runs for 100s with flows randomly starting from 10-15s and ending afterwards till 100s. The frame intervals of these flows are normally distributed in 0.1-1ms. Detailed parameter values for this simulation is shown in Table 4.9.

From Fig. 4.10 we can see that the delivery ratio of STP is significantly worse as compared with other schemes due to its lack of Multipath capability and poor utilization of network bandwidth. The gap between TRILL and DLBMP comes from the difference between static and dynamic algorithms. TRILL implements a static algorithm which assigns a flow sequentially regardless of flow size and current traffic distribution, such that big flows might be assigned to an already congested path while other paths may be lightly loaded. On the other hand, DLBMP assigns flow according to current path load information which can utilize network resource more evenly. In addition, with the introduction of SDN, access switches in DLBMP can exchange path load information effectively with short delay. DLBMP+FFD+CC outperforms TRILL and DLBMP due to FFD and CC mechanisms. FFD offers a fine grained flow assignment that optimizes network bandwidth utilization. Moreover, the integrated CC mechanism can guarantee lossless delivery by successfully controlling data sending rate from sources in an overloaded network. Hence the performance gaps become more obvious after flow number increases to higher than 1900 flows.

Table 4.9: Parameter setting in Experiment 3 and Experiment 4

Name	**Value**
Simulation Time	100s
Flow Number	[1700, 1900, 2100, 2300, 2500]
Threshold T	8Gbps
Cut Ratio b	0.5
Sources	Host1-Host8
Destinations	Host9-Host16
Flow Type	0-199
Start Time	10-15s
End Time	10-100s
Flow Interval	0.1-1ms

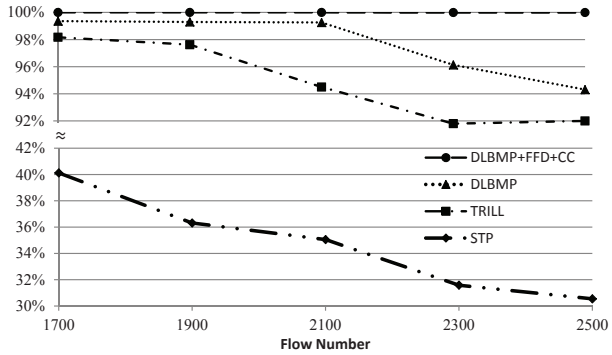


Figure 4.10: Frame delivery ratio VS flow number.

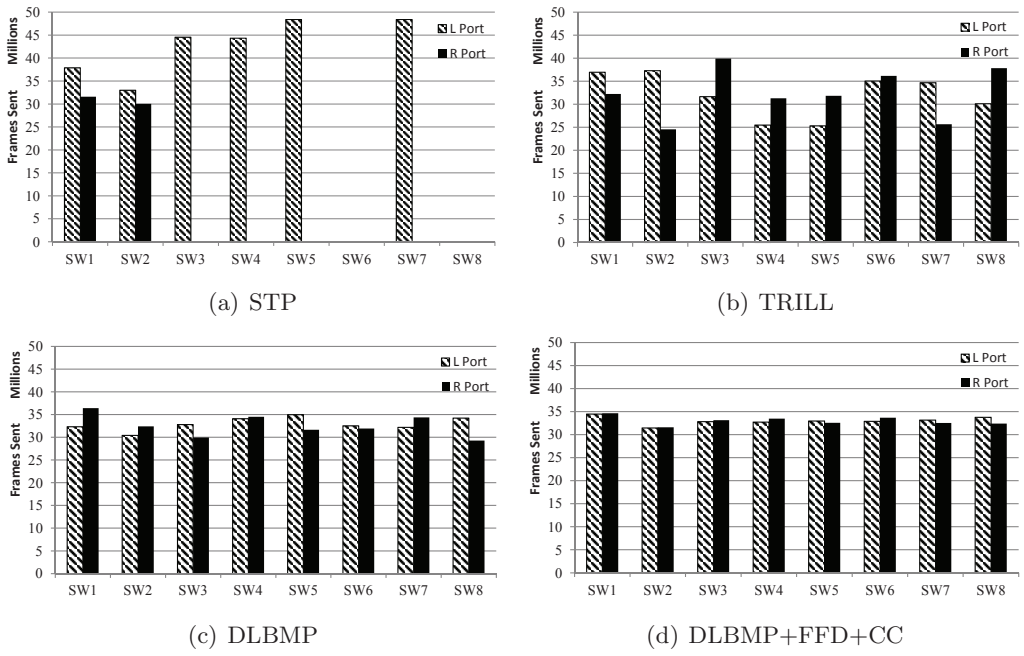
4.3.4 Experiment 4: Load Balance Capability Comparisons

In the fourth experiment, we demonstrate the number of frames sent at each switch to verify the load balance capability of our scheme.

Throughput on each outgoing port is observed in this experiment with the same parameter settings as Experiment 3. We list the statistics collected from the simulation test with 2100 flows. However, they are equally comparable to those tests with other flow numbers.

In Fig. 4.11, we show the total number of frames (in million) sent from all the switch ports towards destinations, Pod3 and Pod4. These ports include uplink ports of SW1, SW2, SW3 and SW4, as well as downlink ports of SW5, SW6, SW7, SW8, SW9 and SW10. To clearly illustrate the ports in the same switch, we name them as L port for those on the left, and R port for those on the right, as appeared in Fig. 4.7. We can see from Fig. 4.11 that traffic load is mostly imbalanced in STP algorithm. TRILL improves significantly as compared to STP. However, our DLBMP and DLBMP+FFD+CC schemes outperform TRILL, which is mainly due to their periodic inspection on path load and dynamic traffic distribution. With a fine grained flow control, DLBMP+FFD+CC balances traffic most evenly among the four schemes with throughput of all ports ranging between 30 to 35 million of frames due to the introduction of FFD.

Besides, DLBMP+FFD+CC also yield a much higher throughput than STP. In STP protocol, traffic is aggregated to L port of SW7, the total throughput only reaches around 48 million frames, equivalent to 101GB. On the other hand, DLBMP+FFD+CC



(a) STP (b) TRILL

(c) DLBMP (d) DLBMP+FFD+CC

Figure 4.11: Throughput at each outgoing port with regard to different algorithms.

can evenly distribute traffic among all ports of egress switches, namely both L and R ports for SW7 and SW8, thus it yields a throughput around 130 million frames, equivalent to 274GB. In this experiment, DLBMP+FFD+CC improves throughput to more than two times as compared to STP. We can estimate that our scheme can gain a higher performance for more traffic volume, since the network capacity has not been fully utilized in this experiment of 2100 flows.

4.4 Summary

In this chapter, we proposed a switch control scheme to provide flow-level multipath capability. We further integrated it with source congestion control. This integrated scheme aims to maximize network throughput as well as prevent frame drop. Simulation results validate our achievements. Firstly, they demonstrated that our proposed switch control scheme can improves the throughput more than two times as compared to STP. Secondly, they also showed that our scheme maintains no frame drop even when traffic volume of 2500 flows exceeds network capacity.

Chapter 5

Link Control for Adapting Congestion

5.1 Motivation

With the increasing scale, data centers have become an energy hog. The energy consumption in data centers has drawn more and more attentions. Currently, data centers are estimated to consume around 120 billion kilowatt hours per year in 2011 [91]. Several efforts have been made to achieve more energy-efficient data centers [92], [93], [94]. Yet the main efforts of these works are concentrated on servers for reduction of energy cost and related cooling. Contrary to the improvement in power savings on servers, power consumption on network remains almost at the same stage.

In Chapter 4, we review several proposals focusing on L2MP and load balancing in data centers. These data center L2MP proposals are attempting to balance traffic all the time. Even under light load, these proposals attempt to involve as many as possible links available. In such a case, network energy consumption remains the same regardless of traffic load.

However, studies on data center traffic characteristics show that network is seldom utilized at its peak capacity. Research works given in [95], [22], [96] reveal the bursty nature of traffic. They reported that the "elephant" flows[1] only last about one second. In contrast, data center network devices and links are running at almost the same level regardless of their traffic loads [76], thus a large part of network energy is wasted in idle

[1] An "elephant" flow means an extremely large (in total bytes) continuous flow.

state. Considering the aforementioned phenomena, proposals with energy proportional network in data centers have been developed including ElasticTree [76] with a central control optimizer, and energy proportional network [22] with an independent control of unidirectional channels on network links. Yet individual control of link rates and feasible traffic aggregation algorithms is still missing in these proposals.

We propose a link control scheme to avoid unnecessary energy usage in data centers by adapting link utilization with network congestion state. The scheme consists two major parts, including *traffic aggregation algorithm* and *adaptive link rate management*. The traffic aggregation algorithm tunes load balance mode, which aggregates traffic to a certain number of links when traffic load is light and distributes traffic to other links when traffic becomes heavier. The adaptive link rate management adjusts link operating rate in accordance with traffic load.

The scheme is proposed based on our observation that redundant links in network architecture take considerable amount of energy consumption. For the purpose of increasing network bandwidth and fault tolerance, redundant links exist in data centers which consume large proportion of powers. Many recently proposed data center architectures such as Fat Tree [3], DCell [4] and VL2 [68] are all equipped with many redundant links. Indeed, the redundancy and load balancing function it provides can prevent SPOF in case of link failure, as well as ensure efficient delivery in case of heavy traffic. However, considering the comparatively light traffic in data center sometimes, traffic aggregation algorithm that offers optional load balancing is a preferable way in saving data center energy.

Besides, data center switches can be configured at several link rates, both studies of adaptive link rate (ALR) [75] and our experiments indicate that different link rates of switch ports consume energy at various levels. As a result, instead of all switch ports running at the highest rate, the adaptive link rate management that dynamically changes link operating rates consumes less power, especially in an under utilized network environment.

To address these issues, we present the link control scheme that adaptively selects part of network topology by activating or deactivating some of redundant links as well as adjusts link rates of switch ports according to traffic loads. Our proposal is based on Fat Tree topology which proposes an attractive architecture for data centers. The rest of the chapter is organized as follows. Section 5.2 introduces background information

by presenting our experiment results for switch power consumption level with different number of links and link rates. We also describe Fat Tree topology with specially defined terms for easier reference in our proposal. Section 5.3 analyzes the network model and formulates an optimization problem. After discussion based on the formulation, we develop a greedy approach. We further test our scheme with OMNET++ simulator and demonstrate results regarding simulation tests with different settings in Section 5.4. A summary is given in Section 5.5. This research work has been summarized and published in [97].

5.2 Background

In this section, we offer background information for our proposals. Firstly, we describe Fat Tree topology focusing on the link construction for energy consumption. Secondly, we introduce the concept of ALR and present our experiment results for validation of these statements.

5.2.1 Fat Tree Architecture

Clos network topology [98] has been proposed half a centenary ago for telephone switches, it aims to deliver high network throughput with commodity switches. Recently, Al-Fares *et al.* adopt a standard instance of a Clos topology called Fat Tree [3]. Fat Tree provides non-blocking network connection with 1:1 oversubscription ratio [3]. We considered Fat Tree topology for our optimization analysis, as Fat Tree topology is widely acknowledged in the field. However, with some adjustments our proposal can also be applied to other topologies.

Details about Fat Tree as a data center architecture can be found in [3]. However, for description completeness, we summarize those related work here. In a data center of a m-ary tree as shown in Fig. 5.1, it is made up by m-port switches, where m should be a positive even number. A Fat Tree contains three layers, namely edge layer, aggregation layer and core layer. The lower two layers are separated into m pods, each pod containing two layers of $m/2$ switches, with lower layer of edge switches and upper layer of aggregation switches. In the upmost layer, there are $(m/2)^2$ core switches, each core switch has a connection to each of the m pods.

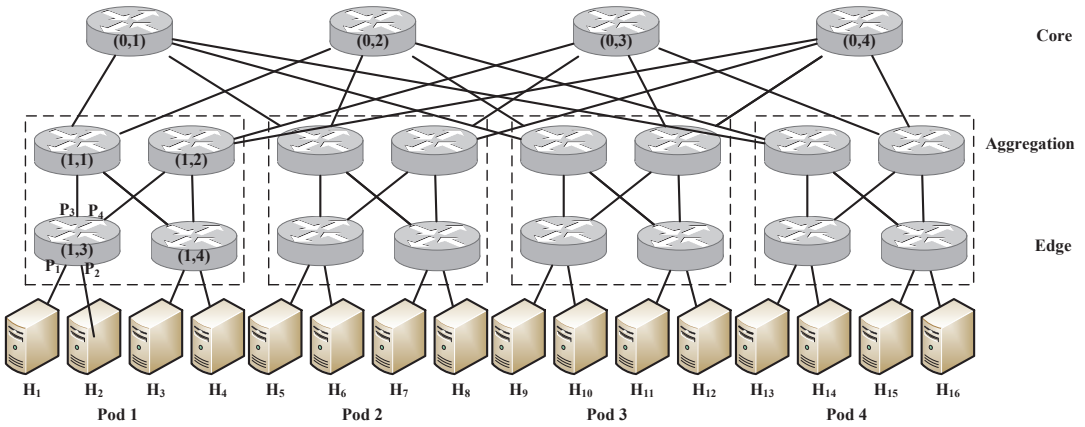


Figure 5.1: Illustration of 4-ary Fat Tree topology.

For clear explanation, we use an ordered pair $\langle p, r\rangle \in \mathcal{R}$ to denote a switch, where set $\mathcal{R}$ is

$$\begin{aligned}\mathcal{R} &= \{\langle 0, r\rangle \,|r \in \{1, 2, ..., (m/2)^2\}\} \\ &\bigcup \{\langle p, r\rangle \,|p \in \{1, 2, ..., m\}, r \in \{1, 2, ..., m\}\}\end{aligned}$$

and p denotes the pod identity of switches from left to right, except $\langle 0, r\rangle$ means core switches, and r denotes the sequence of switches in each pod from left to right and top to bottom, as shown in Fig. 5.1.

The topology follows strict connection rules. For the edge switch, half of the edge switch ports are connected to $m/2$ end hosts and the rest half of switch ports are connected to the $m/2$ aggregation switches in the same pod. In other words, each edge switch has access to each aggregation switch in the same pod and no access to switches in the other pod.

Each pod connects to each core switch through its aggregation switches. Each aggregation switch connects to $m/2$ core switches, each core switch has access to all the pod, port k of each core switch is connected to Pod k. Here, $k \in \{1, 2, ...m\}$ denotes the switch port number, from left to right and top to bottom.

Depending on above discussed connection relationship, we use a matrix function $C(\langle p_l, r_l\rangle, k_l, \langle p_h, r_h\rangle, k_h)$ to denote connections between ${k_l}^{th} \in \{1, 2, ..., m/2\}$ port of lower layer switch $\langle p_l, r_l\rangle$ and ${k_h}^{th} \in \{1, 2, ..., m\}$ port of higher layer switch $\langle p_h, r_h\rangle$,

thus

$$C\left(\langle p_l, r_l\rangle, k_l, \langle p_h, r_h\rangle, k_h\right) = \begin{cases} 1, \mathscr{C} \\ 0, otherwise \end{cases}, \tag{5.1}$$

where the condition $\mathscr{C}$ is defined by

$$\left(p_h = 0, k_h = p_l, r_h = k_l + (r_l - 1) \times \frac{m}{2}\right) \tag{5.2}$$

$$\text{or } \left(p_h = p_l \neq 0, r_h = k_l, k_h = r_l - \frac{m}{2}\right). \tag{5.3}$$

Condition (5.2) specifies connections between core and aggregation switches, in which port k of core switch $\langle 0, r\rangle$ is connected to Pod k, and aggregation switches are connect to core switches with a stride of $m/2$. Condition (5.3) presents connections between aggregation and edge switches, in which the switches are in the same pod and each port connects to a switch in the other layer.

5.2.2 Power Consumption of Switch Links

It has been revealed in [75], [99] that energy consumption varies among different link rates in Cisco Catalyst 2970 switches. However, experiment results are lack of performances of 10G switch links. To report a more timely power consumption rate of switches, we set up a test scenario for Dell PowerConnect 8024F switch as shown in Fig. 5.2 using Hameg HM8021 Multimeter. After power measurement test, we yield power consumption rate with different number of active links as shown in Fig. 5.3.

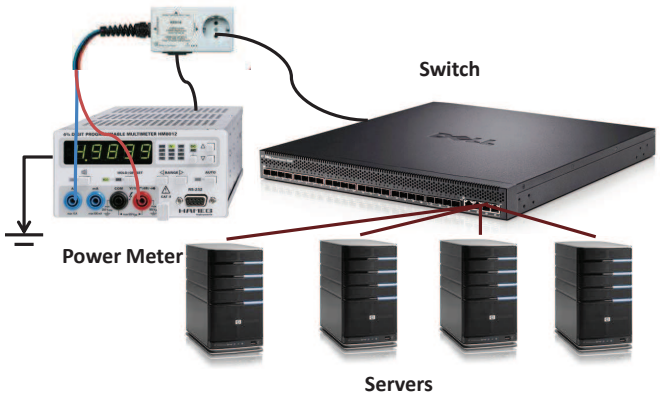


Figure 5.2: Power measuring testbed.

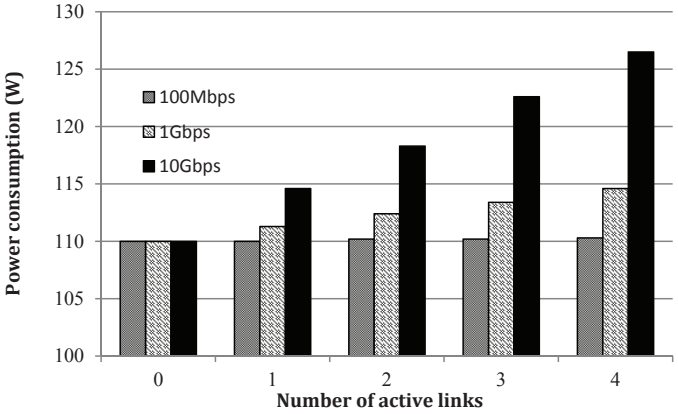


Figure 5.3: Power consumption of Dell PowerConnect 8024F.

In our experiment, the switch power consumption does not vary when traffic load changes. That means, either with no traffic load or full load, a switch consumes the same amount of power given that a certain set of link rates. From this figure, it can be seen that switches consume more power given increasing either link rates or number of links. We also observe that power consumption follows a linear pattern in terms of the number of links. We can roughly estimate that each 100Mbps link consumes no more power than system idle state, while each 1Gbps link consumes around 1.2W more and each 10Gbps link consumes around 4.3W more power, with a base of 110W. When all ports are idle, a switch can switch to SLEEP mode which consumes 10W. In order to verify the linear increment of power, we further test on different settings with a mix of link rates as shown in Table 5.1, which shows the similar results as we expect.

Table 5.1: Switch power consumption of Dell PowerConnect 8024F.

Number of links			Power consumption
100Mbps	1Gbps	10Gbps	(W)
0	1	1	115.6
1	0	2	118.7
0	1	2	119.7
0	2	2	120.9

5.3 Model Formulation and its Solution

In this section we formulate link aggregation model to an optimization problem given the constraint that the output of a port is equal to the input of another port if a connection between the two ports exists. Further, we obtain a greedy approach and present it later in this section.

5.3.1 Optimization Formulation of the Problem

For each switch $\langle p, r\rangle$, $\lambda_{k,d}^{\langle p,r\rangle}$ denotes the arrival load at incoming port k that is destined to destination d. Likewise, $\mu_{k,d}^{\langle p,r\rangle}$ denotes the departure load at outgoing port k that is destined to destination d. The traffic load in each outgoing port also presents as the input traffic load for its adjacent port. That means,

$$\lambda_{k_h,d}^{\langle p_h,r_h\rangle} = \sum_{p_l,r_l,k_l} \mu_{k_l,d}^{\langle p_l,r_l\rangle} \cdot C\left(\langle p_l, r_l\rangle, k_l, \langle p_h, r_h\rangle, k_h\right), \tag{5.4}$$

and

$$\lambda_{k_l,d}^{\langle p_l,r_l\rangle} = \sum_{p_h,r_h,k_h} \mu_{k_h,d}^{\langle p_h,r_h\rangle} \cdot C\left(\langle p_l, r_l\rangle, k_l, \langle p_h, r_h\rangle, k_h\right). \tag{5.5}$$

Our objective is to choose link rates $l_i^{\langle p,r\rangle}$ for each switch, where $i \in \{1, 2, ..., m\}$, so as to minimize the total power consumption $\sum_{p,r} P(\sum_i l_i^{\langle p,r\rangle})$ as follows,

$$\min \sum_{\langle p,r\rangle} P(\sum_i l_i^{\langle p,r\rangle}) \tag{5.6}$$

subject to

$$\sum_k \lambda_{k,d}^{\langle p,r\rangle} = \sum_k \mu_{k,d}^{\langle p,r\rangle}, \tag{5.7}$$

$$\sum_d \lambda_{k,d}^{\langle p,r\rangle} \leq l_k^{\langle p,r\rangle}, l_i \in \mathcal{L}, \tag{5.8}$$

$$\sum_d \mu_{k,d}^{\langle p,r\rangle} \leq l_k^{\langle p,r\rangle}, l_i \in \mathcal{L}, \tag{5.9}$$

for each switch $\langle p, r\rangle$ where $\mathcal{L} = \{100\text{Mbps}, 1\text{Gbps}, 10\text{Gbps}\}$. The constraint in Eqn. (5.7) presents the equality of incoming traffic and outgoing traffic destined to the same end host. Eqn. (5.8) constrains that link rate of a port should be capable of handling all the

incoming traffic on this port. The same condition applies to outgoing traffic as shown in Eqn. (5.9) for full duplex link of switches.

It is easily observed that for packet routing in a Fat Tree topology, packets often first travel to a core switch from the source, and then back to the edge switch connecting the destination. We shall call the packet forwarding from sources to cores as uplink transmission flow and packet forwarding from cores to destinations as downlink transmission flow.

In finding the minimum power consumption, from Eqn. (5.6), it is intuitive that a lesser number of activated links which can adequately carry the incoming traffic will lead to lower power consumption. For the uplink transmission flow, to lower the number of activated links yet supporting the load, the Equations (5.7)-(5.9) suggest the aggregation of traffic flows. That is, the network should use the smallest possible number of output ports for uplink transmission.

Likewise, for the downlink transmission flow, minimizing the activated links should be sought. Based on the Fat Tree topology, we notice that there is always a single path packets can travel from a particular core to its destination. The optimal power consumption is achieved when packets are aggregated during the uplink transmission in a way that packets to the same destination are aggregated to the same core to achieve maximum sharing the downlink transmission. This can be achieved by aggregating packets of the same destination first during the uplink, as specified in Eqn. (5.7).

5.3.2 A Greedy Approach

Based on our observation from the system formulation, we propose a greedy approach which satisfies both requirements indicated by Equations (5.6)-(5.9). The key idea in this solution is for the network to utilize as few switches, switch links and low switch link rates as possible. This link control consists of two mechanisms, namely adaptive link rate management and traffic aggregation algorithm.

Firstly, in adaptive link rate management, a link should aim to run at the lowest possible link rate. In the initial stage, network system begins with no active switches, switches are only enabled when a packet arrives. Packets are automatically routed to a path on a spanning tree with the least link rate given the traffic load. Port buffers are examined for every β packets, links are upgraded or downgraded whenever needed. A link is upgraded whenever potential congestions happen which is indicated by buffer

level exceeding a preset upgrade threshold T_{up}, and downgraded when a relief on the link is observed. To avoid a frequent change of state transition on a port, we only assume a congestion relief when a buffer level has been lower than a preset downgrade threshold T_{down} for a time interval of τ. Secondly, in traffic aggregation algorithm, a switch should aim to use as few links as possible. If any link almost reaches its maximum capacity, its switch will expand the tree structure by opening an adjacent link to support heavier traffic load. Disabling of ports happens in the condition that their buffers have been empty for a certain period of time.

In the implementation, state change of a link is negotiated between its two end ports through DCBX protocol. In DCBX protocol, a link establishes its rate as the higher rate required of its two end ports. Moreover, traffic load will be distributed with a certain quota according to the link rate.

5.3.2.1 Adaptive Link Rate Management

From a port's point of view, adaptive link rate management adapts link rate to traffic load. Each port may experience several states, namely SLEEP, 100Mbps, 1Gbps and 10Gbps. The transitions among these states are performed according to the two thresholds, T_{up} and T_{down}, and an interval time τ. Figure 5.4 illustrates the transitions. That means, a port upgrades to a higher rate if its queue length exceeds the upgrade threshold T_{up}, and downgrades to a lower rate, if its queue length has been lower than the downgrade threshold T_{down} for a period of τ.

However, we adopt a slight different rule for upgrading from or downgrading to SLEEP state. A link is upgraded from the SLEEP whenever a packet arrives and downgraded to SLEEP only when its queue has been empty for τ seconds. In simulation, we create a time trigger whenever queue length goes under T_{down}, which might be

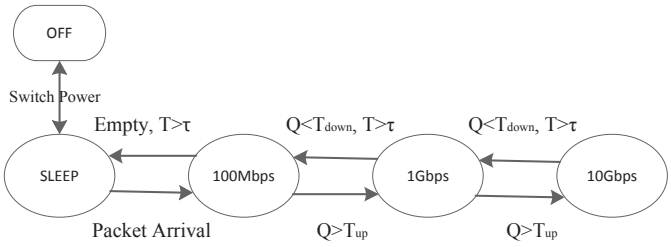


Figure 5.4: Port state transition.

canceled if the queue length rises over T_{down} before τ seconds. Otherwise, the port will downgrade once the trigger enables.

5.3.2.2 Traffic Aggregation Algorithm

From a switch's point of view, traffic aggregation algorithm aggregates traffic to as few as possible links. The algorithm instructs turning on a new link when all of the active links almost reach the current capacity, which is indicated by that all the active links are running at the maximum speed and at least one of their queue lengths exceeds the upgrade threshold. In such a case, the newly turned on link can adapt congestion by sharing load on its neighbor links. In contrary, when the congestion is relief, extra links will downgrade or even turn to SLEEP to aggregate traffic. In other words, Port_x can only be activated if Port_{x-1} reaches maximum rate and Port_x can downgrade only if Port_{x+1} is turned to SLEEP. That means, a link can only be activated or downgraded when all the other active ports are transmitting at the highest link rate. As constrained by this algorithm, there is at most one active link not running at maximum link rate in a switch. Let us take the simple Fat Tree in Fig. 5.1 as an example, it shows that ports in switch $\langle 1, 1\rangle$ are Port_1 to Port_4. By default, when a packet of an uplink transmission flow arrives, the switch will automatically open Port_1. When Port_1 becomes congested with link rate of 10Gbps, Port_2 will be enabled and running at 100Mbps as shown in Fig. 5.5.

As for a switch with $x+1$ active links, there are fully utilized links from Port_1 to Port_x, while Port_{x+1} might run at a rate under 10Gbps. Another issue here is how to distribute loads among flows with a proportion roughly equal to rate ratio among links. This distribution ratio guarantees that the potential port for degradation will not be overflowed by excessive traffic while other ports are free of congestion. To decide

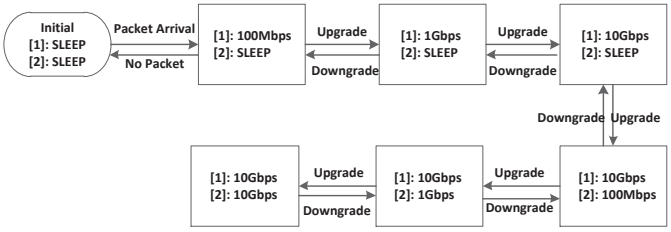


Figure 5.5: Switch state transition.

which port to route to, the mechanism first calculates the total bandwidth of all $x+1$ active links to yield proportion of each port, then it generates a random number which assigns the flow to a certain port. In simulation, this random number is generated by a hash function, which hashes according to flow information. Note here that each activating or deactivating of a port is followed by a rescheduling of flow assignment. We only consider the flow rescheduling without packet reordering. We assume that packet reordering is handled by upper layer protocols such as TCP.

5.4 Experiments and Result Discussions

In this section we present and compare computer simulation results of a common routing algorithm and our scheme. The two algorithms are simulated using OMNET++ 4.1 [84].

To better illustrate, here we use a simple 4-ary Fat Tree as shown in Fig. 5.1. We conduct two simulation experiments, the first shows mechanism testing of our simulation model and the second presents a more realistic scenario with performance evaluations and comparisons. Common parameters are listed in Table 5.2.

Table 5.2: Common parameter setting.

Name	Value
State changing interval τ	0.3s
Buffer checking interval β	20 packets
Upgrade Threshold T_{up}	50 packets
Downgrade Threshold T_{down}	1 packet

5.4.1 Experiment 1: Single Flow Test

We firstly run a simulation experiment with a flow generated from H_4 to H_8. The flow starts at 1s and ends at 6s. In between, the flow frequency changes on every second, which starts with a packet generating interval of 10ms at 1s, grows to 0.2ms at 2s and 0.01ms at 3s, then returns to 0.2ms at 4s and 10ms at 5s, and finally stops at 6s. The simulation parameters are listed in Table 5.3. Figure 5.6 shows the throughput variations of the system which presents a clearer idea of the frequency changes.

Table 5.3: Simulation setting in Experiment 1.

Name	Value
Number of flows	1
Sender	H_4
Receiver	H_8
Flow start time	1s
Flow end time	6s

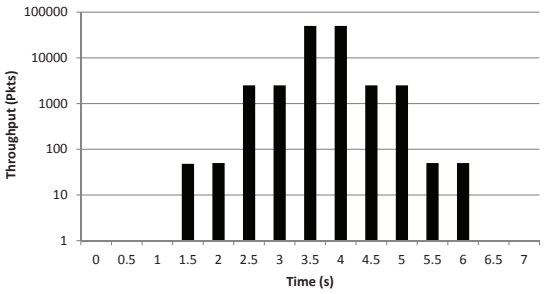


Figure 5.6: Throughput at H_8.

During the simulation, we can see the flow taking the route $H_4 \rightarrow \langle 1,4\rangle \rightarrow \langle 1,1\rangle \rightarrow \langle 0,1\rangle \rightarrow \langle 2,1\rangle \rightarrow \langle 2,4\rangle$, whose links increase rate as the flow frequency climbs and reduce rate when the flow frequency drops. Here we present statistics of switch state which records time spent on each state of aggregation switch $\langle 1,1\rangle$ in Fig. 5.7.

In this figure, we denote switch state in an $[a,b,c,d]$-tuple, in which a denotes number of ports in SLEEP mode, b,c,d denote number of ports running at 10Mbps, 1Gbps and 10Gbps respectively. For example, $[1,1,0,2]$ presents the state when there

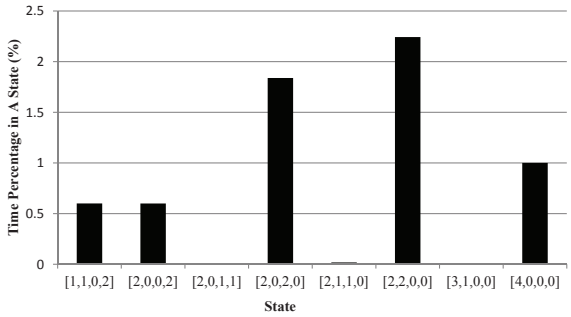


Figure 5.7: Switch state time for aggregation switch $\langle 1,1\rangle$.

are one link in SLEEP mode, one link at 10Mbps and two links at 10Gbps in a four-port switch. From this figure, we see that rather than all ports running at 10Gbps rate in a common data center network, in our scheme switch $\langle 1, 1\rangle$ changes its state to a lower power consumption mode which yields a less power consumption along the way. In Fig. 5.7, we do not present those states maintaining an almost zero state time.

5.4.2 Experiment 2: Multiple Flows Test

Secondly, we conduct a series of simulation tests with a more general and realistic settings in this experiment. We increase number of flows to present a general system performance at the level of a data center throughput capacity. In these tests, traffic is randomly generated from all of the hosts, H_1 to H_{16}, to all other hosts, with flow frequencies (denoted in terms of packet generation interval) from 0.1ms to 1ms. These flows start randomly from 1s to 6s, and the end time randomly falls into the range in between of their start time to 15s. Our simulation results include the tests on 10, 20, 30, 50, 100, 200 and 300 flows. The simulation parameters are listed in Table 5.4. In these simulation tests, we firstly focus on calculating the total energy saving for described system given different flow numbers. Secondly, we obtain delay statistics to provide a guideline as well as justify the feasibility of our scheme.

According to switch power consumption we measure in Section 5.2, we can theoretically estimate total energy consumption in our system, based on the switch state statistics yielded by simulations. We present the energy usage results in Table 5.5 and Fig. 5.8 in comparison with a common Fat Tree implementation. The results here are obtained through a procedure to achieve accuracy of calculation. For each number of flows, we first collect the switch state information, sum the time spent in each switch

Table 5.4: Simulation setting in Experiment 2.

Name	Value
Number of flows	10, 20, 30, 40, 50, 100, 200, 300
Sender	H_1-H_{16}
Receiver	H_1-H_{16}
Flow start time	1s-6s
Flow end time	flow's start time to 15s

Table 5.5: Energy usage comparison.

Number of flows	**Energy usage (J)**	
	Our Scheme	**Fat Tree**
10	14009	37800
20	21844	36540
30	21115	37800
40	22274	36540
50	21786	36540
100	26250	37800
200	29119	37800
300	29585	37800

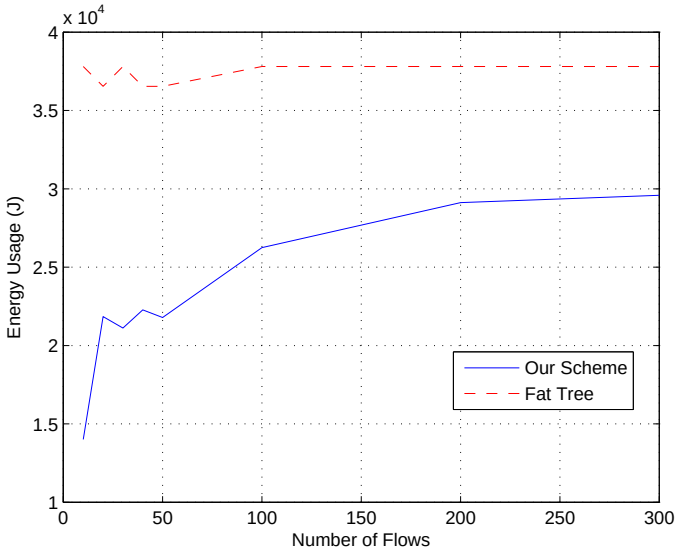


Figure 5.8: Illustration of Energy Usage Comparison.

state for each switch in the system, and further integrate states that have the same power consumption. Then we calculate the total power consumption according to power measurement on Dell PowerConnect 8024F switch from the discussions in Section 5.2.

The results we obtained here show that the overall power consumption increases as the number of flows increases, the mere inconsistency may due to unbalance flow distribution from random generation. We see there is no difference on power consumption for

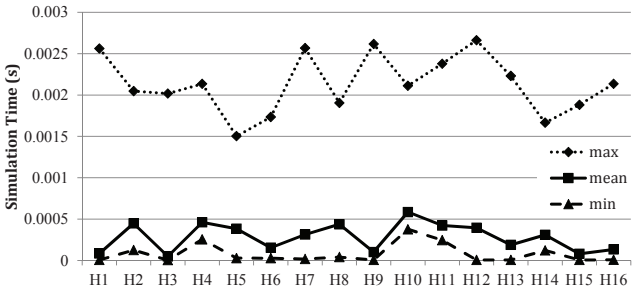


Figure 5.9: Max, mean and min End-to-End delay of 50 flows.

SLEEP mode and 100Mbps on a port in the aforementioned Dell model, and for other data rates, upgrading also does not intrigue significant power change. However, preliminary measurement on other models varies more in power consumption for different link rates, hence our results that use Dell switch represent the conservative measurement among all. Based on the results, we confirm the effectiveness of our method even with conservative measurement. We would like to mention here, the difference of energy usage in Table 5.5 for 20, 40, 50 flows comes from the randomly ending of flows in which all flows end before 15s. The earlier ending explains the energy usage of Fat Tree in 20, 40 and 50 flows tests are less than that in the other tests.

Besides, we also study latency in deploying our proposed scheme, since the switching of ports and port rates may result in a longer delay. In Fig. 5.9, we show the delay statistics of 50 flows. The end-to-end delay is generally maintained at a very low value. Even the highest delay time in this simulation is considered as acceptable as discussed in [75].

Figure 5.10 shows latency statistics which give average latencies for each host with each number of flows. From the figure, we can see that the end-to-end delay is within a low and constrained range even with heavier traffic load.

From the above simulation results, we conclude that using our proposed scheme, the mean delay among all hosts tends to be less as the number of flows increases. This is mainly due to the increase of link rate as traffic increases, thus results in faster transmission and lesser latencies. On the other hand, a higher traffic load means higher stability of operation, as there are less state transitions and traffic is more evenly distributed. Both of them account for a smaller and more evenly distributed delay with more number of flows.

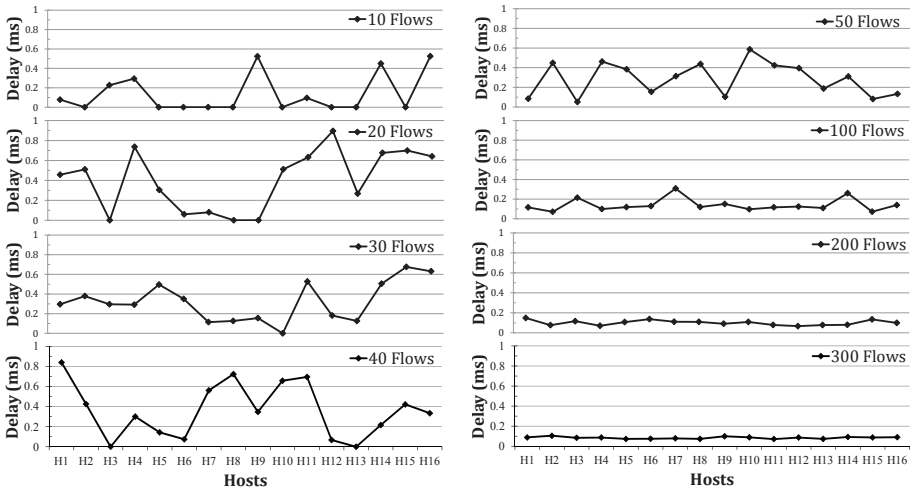


Figure 5.10: Hosts delay statistics.

5.5 Summary

In this chapter, we presented the formulation and solution of our work as a novel link control scheme for energy efficient data center Ethernet. The design principle of our scheme was to explore a proper number of active links and link rate levels for adaptation with network congestions. By measuring the real power consumption of a switch model and implementing simulation test, we estimated power saved in our proposed scheme. In the simulations we observed a saving of 60% in light traffic load, while we did not see much delay incurred by our algorithm, and delay time became shorter with increasing number of flows.

Chapter 6

Conclusion

6.1 Research Contribution

Ethernet is a primary link-layer protocol for LAN connections, as it is economical for deployment, easy for management and commercially available for decades. As its transmission rate speeds up to 10Gbps, even the development of 40Gbps and 100Gbps in a near future, it has been considered as a potential candidate for data center I/O consolidation, owing to its credibility on simple implementation, low cost and few management requirements. Yet it remains inadequate as a unified fabric for data centers on prioritized congestion control, maximal throughput congestion easing, and optimal power congestion adaptation.

Conventionally, Ethernet is designed to be a best-effort traffic network that may drop frames in appearance of buffer overflow. Since it carries only IP traffic, there is no differentiation on traffic with priorities. However, with the introduction of other traffic classes such as FC or IPC traffic, Ethernet faces challenges on protecting this loss sensitive traffic from drop as well as prioritizing them to satisfy their throughput and latency requirements. Moreover, Ethernet LAN runs STP protocol that reduces network redundancies to a spanning tree structure to prevent loop. That means, only a single path exists between each source-destination pair. Although it curbs network broadcast storm from happening, the pruned structure significantly limits network bandwidth, which further effects overall throughput. As all traffic flows are traveling through a single tree, the network becomes more prone to congestions. Thus, an L2MP scheme is in demand to increase throughput, distribute traffic and ease congestions.

Additionally, the use of link redundancies, although expanding network capacity, incurs unnecessary power consumption under light traffic load. Tradeoffs often exist between load balance and aggregation, or among various link rates, which presents another issue of link adaptation on congestions to optimize energy consumption in data centers. Inspired by these observations, we conducted research to enhance Ethernet congestion management under data center environment. Three schemes are proposed including source control for throttling congestion, switch control for easing congestion and link control for adapting congestion. The complete solution of these three schemes satisfy congestion management requirements of a prioritized, high throughput and energy-efficient lossless data center Ethernet.

In the first scheme, we discovered that several attempts have been made to tackle the two requirements of lossless transmission and priority differentiation separately in literature. Nevertheless, existing proposals lack fine granularity differentiation over various priorities. Independent handling towards these two requirements yields unpredictable results when incorporate them together in real systems, so that it is risky to achieve these two requirements respectively. We specifically designed a cooperative source control scheme and performed in-depth study on our scheme to investigate thoroughly the stability and suitability for such a prioritized congestion control. By analyzing its throughput ratios over various priorities, we showed that AQM and AIMD technique is suitable for solving data center differentiation congestion control problem. Moreover, guidelines for system parameter selection were provided. Given a required throughput ratio, the system would be stable and its overhead could be minimized. In our simulation experiments, we demonstrated that our design has favorable performance on both avoiding congestion and prioritizing traffic classes over other candidates. This research work has been summarized and published in [80], [81].

Secondly, from literature review, L2MP algorithms have been mainly classified into oblivious and adaptive categories. Oblivious routing algorithms route packets to all available paths between each source-destination pair equally with no regard of existing traffic load. Most of the current proposals fall into this category since it is simple to implement and analyze. However, to obtain optimal network throughput, an oblivious approach often requires evenly distributed traffic pattern; for other traffic patterns, the potential performance can be hardly achieved. An adaptive routing algorithm adopts information about network state. Although including information about network state

can potentially improve routing performance, an adaptive approach usually adds considerable complexity. If not designed carefully, the overhead can lead to performance degradation. The above observations motivate us to propose a design that achieves the best of two worlds. Utilizing SDN technology, routing control information can be directly exchanged through a central controller without traveling through common data paths. Through separation of intelligence, we minimized network complexity, and thus avoided performance degradation. Since routing depends on network state, our proposal was intimately coupled with congestion control mechanism which preserves a network to be lossless. Extensive simulations have been conducted to examine multipath capability and network capacity. The results also showed that even beyond its capacity, a network could maintain lossless due to effective congestion control. This research work has been summarized and published as in [87], [88].

Lastly, we are interested in energy proportional data center which is at odds with evenly load balance. We observed that every route is activated in a load balanced network, even though traffic in data centers follows a bursty nature that most of the time the load appears light. Considering this observation, we proposed to adaptively activate or deactivate part of network links and adjust link rates of switch ports according to congestion level. In particular, we formulated the relationship between power consumption and traffic load, and proposed a greedy algorithm to find the tradeoff between load aggregation and distribution. In the network level, partial topology can be switched to sleep mode during light load to minimize switch usage. In the switch level, we studied the impact on power consumption of number of active links and link rates for several switch models. From this study, we discovered that a switch consumes less energy when operating at a lower rate and with fewer links. Combining these two approaches together, the dynamic range of energy usage was extended, such that it can yield extra savings when network is light-loaded. This research work has been summarized and published in [97].

6.2 Future Work

Data center Ethernet, as a potential candidate for data center consolidation, has received more and more attention for data center performance improvement. All our proposed schemes have enhanced Ethernet performance to be more suitable for data

centers. However, there are still some challenges for data centers with extensive scaling speed, where the number of end stations increases and network topology changes accordingly. Scalability and fault-tolerance are considered as two important metrics to evaluate the performance of a data center solution. To include scalability consideration in future proposals can be one of the research directions.

With respect to each of the three control schemes, current proposals still possess some limitations. Accordingly, promising research directions can be seen from them. From source control perspective, the current proposal considers only throughput as a criterion for priority differentiation. More criteria such as latency, bit error rate and out-of-order delivery may be taken into consideration. Facilitating with these aspects, the differentiated handling of prioritized traffic will become more effective. From switch control perspective, the routing table kept in access switches can be large as data center expands. A more efficient approach to store routing tables and determine routes is in need. Another research direction is to provide fault-tolerant capability in routing. As data centers provide robust and resilient infrastructure, maintaining data accessibility in case of link or switch failure create an important feature of data centers. From link control perspective, the current proposal only considers link rate and link number. From our observation, data center traffic shows an asymmetric distribution between upstream and downstream paths. In such a traffic distribution, simplex or half-duplex mode might be capable to handle traffic transmission. These simpler modes may have a potential to consume less energy than the full-duplex mode. Hence, an algorithm that can switch between duplex modes of a link probably may yield further savings.

Additionally, network statistics accumulated by a central controller is a key enabler for improving network performance. By integrating our solution with network statistics or other technologies such as data mining, smart grid, we are able to offer more efficient load-aware or energy-aware algorithms with better performance or towards a broad scope. Exploring the possibility of incorporating such technologies into the framework forms another direction of future research.

References

[1] C V Hollot, V. Misra, D. Towsley, and W. B. Gong. **A control theoretic analysis of RED**. In *proc. of IEEE INFOCOM*, **3**, pages 1510–1519, 2001. iv, 46, 47

[2] J. Kim, J. Balfour, and W. Dally. **Flattened butterfly topology for on-chip networks**. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 172–182. IEEE Computer Society, 2007. vii, 16, 17, 18

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. **A scalable, commodity data center network architecture**. In *ACM SIGCOMM Computer Communication Review*, **38**, pages 63–74. ACM, 2008. vii, 18, 19, 33, 88, 89

[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. **Dcell: a scalable and fault-tolerant network structure for data centers**. In *ACM SIGCOMM Computer Communication Review*, **38**, pages 75–86. ACM, 2008. vii, 16, 19, 20, 88

[5] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. **BCube: a high performance, server-centric network architecture for modular data centers**. In *ACM SIGCOMM Computer Communication Review*, **39**, pages 63–74. ACM, 2009. vii, 16, 19, 20

[6] **IEEE. 802.1Qbb - Priority-based Flow Control**. Available at `http://www.ieee802.org/1/pages/802.1bb.html`, 2011. vii, 22

[7] **IEEE. 802.1Qaz - Enhanced Transmission Selection**. Available at `http://www.ieee802.org/1/pages/802.1az.html`, 2011. vii, 23, 28

[8] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson. **RAID: High-performance, reliable secondary storage**. *ACM Computing Surveys (CSUR)*, **26**(2):145–185, 1994. 3, 10

[9] R.J. Baker. *CMOS: Circuit design, layout, and simulation*. John Wiley & Sons, Inc., 2011. 3

[10] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, and W.K. Su. **Myrinet: A gigabit-per-second local area network**. *Micro, IEEE*, **15**(1):29–36, 1995. 5

[11] Fibre Channel Industry Association. **Fibre Channel over Ethernet in the Data Center: An Introduction**. http://www.fibrechannel.org/OVERVIEW/FCIA_SNW_FCoE_WP_Final.pdf, 2007. 6

[12] Mellanox Technologies. **The Case for InfiniBand over Ethernet**, 2008. 7

[13] **IEEE. 802.1 - Data Center Bridging Task Group**. Available at http://www.ieee802.org/1/pages/dcbridges.html. 7, 15

[14] G. Smith et al. **Converged Enhanced EthernetGood for iSCSI SANs**. *NetApp White Paper, Blade Network Technologies*, pages 1–7, 2008. 7

[15] K. Nibhanupudi and R. Devgan. **Data Center Ethernet**. 7

[16] J.R. Santos, Y. Turner, and G. Janakiraman. **End-to-end congestion control for InfiniBand**. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, **2**, pages 1123–1133. IEEE, 2003. 8

[17] **IEEE. 802.1Q-2005 Virtual Bridged Local Area Networks**. Available at http://www.ieee802.org/1/pages/802.1Q.html. 8, 31

[18] **IEEE. 802.1D MAC Bridges**. http://www.ieee802.org/1/pages/802.1D-2003.html. 8

[19] **IEEE. 802.1w - Rapid Reconfiguration of Spanning Tree**. Available at http://www.ieee802.org/1/pages/802.1w.html. 8, 31

[20] S. Gai. **Data Center Networks and Fibre Channel over Ethernet (FCoE)**. 2008. 16, 65, 66

[21] S. Gai, C. DeSanti, and C. Press. **I/O Consolidation in the Data Center**. 2010. 16, 21, 65, 66

[22] D. Abts, M.R. Marty, P.M. Wells, P. Klausler, and H. Liu. **Energy proportional datacenter networks**. *ACM SIGARCH Computer Architecture News*, **38**(3):338–347, 2010. 18, 37, 38, 87, 88

[23] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. **PortLand: a scalable fault-tolerant layer 2 data center network fabric**. In *ACM SIGCOMM Computer Communication Review*, **39**, pages 39–50. ACM, 2009. 18

[24] W.J. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004. 20, 33

[25] **IEEE. 802.3 - Eternet Working Group**. Available at `http://www.ieee802.org/3/`. 21

[26] GAI Silvano and D. Claudio. **I/O consolidation in the data center. A complete guide to data center Ethernet & fibre channel over Ethernet**. 2009. 21, 68, 75

[27] **IEEE. 802.1p Traffic Class Expediting and Dynamic Multicast Filtering (published in 802.1D-1998)**. `http://www.ieee802.org/1/pages/802.1D.html`. 22

[28] **IEEE. 802.1Qau - Congestion Notification**. Available at `http://www.ieee802.org/1/pages/802.1au.html`, 2010. 23

[29] S. Floyd, M. Handley, and J. Padhye. **A comparison of equation-based and AIMD congestion control**. Manuscript available at `http://www.aciri.org/floyd/papers.html`, 2000. 24

[30] F. Baccelli and D. Hong. **The AIMD model for TCP sessions sharing a common router**. In *proc. of 39th Annual Allerton Conference on Communication, Control and Computing*, pages 900–911, 2001. 24

[31] Aditya Akella, Srinivasan Seshan, Richard Karp, Scott Shenker, and Christos Papadimitriou. **Selfish behavior and stability of the Internet: a game-theoretic analysis of TCP**. In *proc. of ACM SIGCOMM*, **32**, pages 117–130, 2002. 24

[32] G. Hasegawa, K. Kurata, and M. Murata. **Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet**. In *proc. of International Conference on Network Protocols*, pages 177–186, 2000. 24

[33] Y. R. Yang and S. S. Lam. **General AIMD congestion control**. In *proc. of International Conference on Network Protocols*, pages 187–198, 2000. 24

[34] C. P. Fu, C. H. Foh, C. T. Lau, Z. Man, and B. S. Lee. **Semi-Markov modeling for bandwidth sharing of TCP connections with asymmetric AIMD congestion control**. In *proc. of IEEE Global Telecommunications Conference*, pages 2014–2018. IEEE, 2007. 24

[35] D. Bergamasco and D. Dutt. **Backward congestion notification**, August 21 2007. US Patent App. 11/842,866. 25

[36] D. Bergamasco. **Ethernet congestion manager (ECM) specification**. *Cisco Systems, initial draft EDCS-574018*, 2007. 25

[37] R. Pan, B. Prabhakar, and A. Laxmikantha. **Qcn: Quantized congestion notification**, 2007. 25

[38] Jinjing Jiang, Raj Jain, and Saint Louis. **Forward Explicit Congestion Notification (FECN) for Datacenter Ethernet Networks**. pages 1–40, 2007. 25

[39] C. So-In, R. Jain, and J. Jiang. **Enhanced Forward Explicit Congestion Notification (E-FECN) Scheme for Datacenter Ethernet Networks**. In *Performance Evaluation of Computer and Telecommunication Systems, 2008. SPECTS 2008. International Symposium on*, pages 542–546. IEEE, 2008. 25

[40] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. **Data center tcp (dctcp)**. In *proc. of the ACM SIGCOMM*, pages 63–74, 2010. 25, 51

[41] D. Bergamasco. **Data center ethernet congestion management: Backward congestion notification**. In *IEEE 802.1 Meeting*, 2005. 26, 28, 51

[42] **IEEE 802.1AB - Station and Media Access Control Connectivity Discovery**. Available at `http://www.ieee802.org/1/pages/802.1ab.html`. 28

[43] **IEEE 802.1AB-2009 - Station and Media Access Control Connectivity Discovery**. Available at `http://www.ieee802.org/1/pages/802.1AB-2009.html`. 28

[44] **IEEE. 802.1aq - Shortest Path Bridging**. Available at `http://www.ieee802.org/1/pages/802.1aq.html`, 2012. 29

[45] **IEEE. 802.1ad - Provider Bridges**. Available at `http://www.ieee802.org/1/pages/802.1ad.html`, 2008. 29

[46] **802.1ah - Provider Backbone Bridges**. Available at `http://www.ieee802.org/1/pages/802.1ah.html`, 2008. 29

[47] J. Moy. **OSPF version 2**. 1997. 29

[48] D. Thaler and C. Hopps. **RFC 2991**, 2000. 29

[49] C.E. Hopps. **RFC 2922-Analysis of an equal-cost multi-path algorithm**. Available at `http://tools.ietf.org/html/rfc2992`, 2000. 29

[50] R. Perlman. **Rbridges: transparent routing**. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, **2**, pages 1211–1218. IEEE, 2004. 29

[51] E.W. Dijkstra. **A note on two problems in connexion with graphs**. *Numerische mathematik*, **1**(1):269–271, 1959. 30

[52] J.A. Azevedo, M.E.O. Santos Costa, J.J.E.R. Silvestre Madeira, and E.Q. Vieira Martins. **An algorithm for the ranking of shortest paths**. *European Journal of Operational Research*, **69**(1):97–106, 1993. 30

[53] N. Katoh, T. Ibaraki, and H. Mine. **An efficient algorithm for k shortest simple paths**. *Networks*, **12**(4):411–427, 1982. 30

[54] J.Y. Yen. **Finding the k shortest loopless paths in a network**. *management Science*, pages 712–716, 1971. 30

[55] N. Katoh, T. Ibaraki, and H. Mine. **An O (Kn2) algorithm for K shortest simple paths in an undirected graph with nonnegative arc length**. *Trans. Inst. Electronics and Communication Engineers of Japan E*, **61**:971–972, 1978. 30

[56] B. Jabbari, S. Gong, and E. Oki. **On constraints for path computation in multi-layer switched networks**. *IEICE transactions on communications*, **90**(8):1922–1927, 2007. 30

[57] **IEEE. 802.1s - Multiple Spanning Trees**. Available at `http://www.ieee802.org/1/pages/802.1s.html`. 31

[58] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. **Viking: A multi-spanning-tree Ethernet architecture for metropolitan area and cluster networks**. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, **4**, pages 2283–2294. IEEE, 2004. 31

[59] C. Villamizar. **OSPF Optimized Multipath OSPF-OMP**. 31

[60] P. Narvaez, K.Y. Siu, and H.Y. Tzeng. **Efficient Algorithms for Multi-Path Link-State Routing**. 1999. 31

[61] H.S. Palakurthi. **Study of multipath routing for qos provisioning**. *EECS*, 2001. 32

[62] S. Vutukury and J.J. Garcia-Luna-Aceves. **MDVA: A distance-vector multipath routing protocol**. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, **1**, pages 557–564. IEEE, 2001. 32

[63] S. Vutukury and JJ Garcia-Luna-Aceves. **A Simple Approximation to Minimum-Delay Routing**. In *proc. of ACM SIGCOMM*, Sept 1999. 32

[64] S. Mao, D. Bushmitch, S. Narayanan, and S.S. Panwar. **MRTP: a multiflow real-time transport protocol for Ad Hoc networks**. *Multimedia, IEEE Transactions on*, **8**(2):356–369, 2006. 32

[65] Y. Lee, I. Park, and Y. Choi. **Improving TCP performance in multipath packet forwarding networks**. *Journal of Communications and Networks*, **4**(2):148–157, 2002. 32

[66] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. **Improving datacenter performance and robustness with multipath TCP**. *SIGCOMM-Computer Communication Review*, **41**(4):266, 2011. 32

[67] A. Greenberg, P. Lahiri, D.A. Maltz, P. Patel, and S. Sengupta. **Towards a next generation data center architecture: scalability and commoditization**. In *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pages 57–62. ACM, 2008. 34

[68] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, and S. Sengupta. **VL2: a scalable and flexible data center network**. *ACM SIGCOMM Computer Communication Review*, **39**(4):51–62, 2009. 34, 88

[69] C. Minkenberg, M. Gusat, and G. Rodriguez. **Adaptive Routing in Data Center Bridges**. In *17th IEEE Symposium on High Performance Interconnects*, pages 33–41. IEEE, 2009. 34

[70] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. **Hedera: Dynamic flow scheduling for data center networks**. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pages 19–19. USENIX Association, 2010. 34

[71] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. **OpenFlow: enabling innovation in campus networks**. *ACM SIGCOMM Computer Communication Review*, **38**(2):69–74, 2008. 35

[72] **IEEE P802.3az Energy Efficient Ethernet**. Available at `http://www.ieee802.org/3/az/public/index.html`. 36

[73] M. Gupta and S. Singh. **Greening of the Internet**. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26. ACM, 2003. 36

[74] **IEEE 802.3 LAN/MAN CSMA/CD Access Method.** Available at `http://standards.ieee.org/getieee802/802.3.html`. 37

[75] C. Gunaratne, K. Christensen, and B. Nordman. **Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed**. *International Journal of Network Management*, **15**(5):297–310, 2005. 37, 88, 91, 101

[76] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. **ElasticTree: Saving energy in data center networks**. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pages 17–17. USENIX Association, 2010. 37, 38, 87, 88

[77] Y. Ariba, F. Gouaisbaut, and Y. Labit. **Feedback control for router management and TCP/IP network stability**. *IEEE Transactions on Network and Service Management*, **6**(4):255–266, 2009. 42

[78] F. Paganini, Z. Wang, S. H. Low, and J. C. Doyle. **A new TCP/AQM for stable operation in fast networks**. In *proc. of IEEE INFOCOM*, pages 96–105, 2003. 42

[79] V. Misra, W. B. Gong, and D. Towsley. **Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED**. In *proc. of ACM SIGCOMM*, **30**, pages 151–160, 2000. 42

[80] S. Fang, C.H. Foh, and K.M.M. Aung. **Differentiated Ethernet Congestion Management for Prioritized Traffic**. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE. 42, 46, 104

[81] S. Fang, C. Foh, and K. Aung. **Differentiated Congestion Management of Data Traffic for Data Center Ethernet**. *Network and Service Management, IEEE Transactions on*, (99):1–12. 42, 104

[82] **Converging SAN and LAN Infrastructure with Fibre Channel over Ethernet**. Available at `http://download.intel.com/support/network/sb/ciscointelfcoewhitepaper.pdf`. 42

[83] Y. Lu, R. Pan, B. Prabhakar, D. Bergamasco, V. Alaria, and A. Baldini. **Congestion control in networks with no congestion drops**. In *proc. of 44th Annual Allerton Conference on Communication, Control, and Computing*, pages 891–898, 2006. 47

[84] **OMNeT++**. `http://www.omnetpp.org`. 50, 80, 97

[85] J. Jiang and R. Jain. **Analysis of backward congestion notification (BCN) for ethernet in datacenter applications**. In *proc. of IEEE INFOCOM*, pages 2456–2460, 2007. 50

[86] M. Alizadeh, A. Javanmard, and B. Prabhakar. **Analysis of DCTCP: Stability, Convergence, and Fairness**. In *proc. of the ACM SIGMETRICS*, 2011. 53

[87] S. Fang, Y. Yu, C. Foh, and K. Aung. **A Loss-free Multipathing Solution for Data Center Network using Software-defined Networking Approach**. *to appear in APMRC*. 67, 105

[88] S. Fang, Y. Yu, C. Foh, and K. Aung. **A Loss-free Multipathing Solution for Data Center Network Using Software-defined Networking Approach**. *to appear in Transactions on Magnetics*. 67, 105

[89] T.W. Chim, K.L. Yeung, and K.S. Lui. **Traffic distribution over equal-cost-multi-paths**. *Computer Networks*, **49**(4):465–475, 2005. 76

[90] S. Kandula, D. Katabi, S. Sinha, and A. Berger. **Dynamic load balancing without packet reordering**. *ACM SIGCOMM Computer Communication Review*, **37**(2):51–62, 2007. 76

[91] U.S. Department of Energy. **Data Center Energy Consumption Trends**. `http://www1.eere.energy.gov/femp/program/dc_energy_consumption.html`, 2010. 87

[92] R. Bianchini and R. Rajamony. **Power and energy management for server systems**. *Computer*, **37**(11):68 – 76, nov. 2004. 87

[93] E. Pakbaznia and M. Pedram. **Minimizing data center cooling and server power costs**. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, pages 145–150. ACM, 2009. 87

[94] D. Meisner, B.T. Gold, and T.F. Wenisch. **PowerNap: eliminating server idle power**. *ACM Sigplan Notices*, **44**(3):205–216, 2009. 87

[95] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. **The nature of data center traffic: measurements & analysis**. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 202–208. ACM, 2009. 87

[96] T. Benson, A. Anand, A. Akella, and M. Zhang. **Understanding data center traffic characteristics**. *ACM SIGCOMM Computer Communication Review*, **40**(1):92–99, 2010. 87

[97] S. Fang, H. Li, C.H. Foh, Y. Wen, and K.M.M. Aung. **Energy Optimizations for Data Center Network: Formulation and its Solution**. In *GLOBECOM 2012, 2012 IEEE Global Telecommunications Conference*, pages 1–5. IEEE. 89, 105

[98] C. Clos. **A study of non-blocking switching networks**. *Bell System Technical Journal*, **32**(2):406–424, 1953. 89

[99] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J.A. Maestro. **IEEE 802.3 az: the road to energy efficient ethernet**. *Communications Magazine, IEEE*, **48**(11):50–56, 2010. 91