ABSTRACT

CONGESTION CONTROL, ENERGY EFFICIENCY AND VIRTUAL MACHINE PLACEMENT FOR DATA CENTERS

by
Yan Zhang

Data centers, facilities with communications network equipment and servers for data processing and/or storage, are prevalent and essential to provide a myriad of services and applications for various private, non-profit, and government systems, and they also form the foundation of cloud computing, which is transforming the technological landscape of the Internet. With rapid deployment of modern high-speed low-latency large-scale data centers, many issues have emerged in data centers, such as data center architecture design, congestion control, energy efficiency, virtual machine placement, and load balancing.

The objective of this thesis is multi-fold. First, an enhanced Quantized Congestion Notification (QCN) congestion notification algorithm, called fair QCN (FQCN), is proposed to improve rate allocation fairness of multiple flows sharing one bottleneck link in data center networks. Detailed analysis on FQCN and simulation results is provided to validate the fair share rate allocation while maintaining the queue length stability. Furthermore, the effects of congestion notification algorithms, including QCN, AF-QCN and FQCN, are investigated with respect to TCP throughput collapse. The results show that FQCN can significantly enhance TCP throughput performance, and achieve better TCP throughput than QCN and AF-QCN in a TCP Incast setting.

Second, a unified congestion detection, notification and control system for data center networks is designed to efficiently resolve network congestion in a uniform solution and to ensure convergence to statistical fairness with “no state” switches simultaneously. The architecture of the proposed system is described in detail and the FQCN algorithm is implemented in the proposed framework. The simulation results of the FQCN algorithm implemented in the proposed framework validate the robustness and efficiency of the proposed congestion control system.

Third, a two-level power optimization model, namely, Hierarchical EneRgy Optimization (HERO), is established to reduce the power consumption of data center networks by switching off network switches and links while still guaranteeing full connectivity and maximizing link utilization. The power-saving performance of the proposed HERO model is evaluated by simulations with different traffic patterns. The simulation results have shown that HERO can reduce the power consumption of data center networks effectively with reduced complexity.

Last, several heterogeneity aware dominant resource assistant heuristic algorithms, namely, dominant residual resource aware first-fit decreasing (DRR-FFD), individual DRR-FFD (iDRR-FFD) and dominant residual resource based bin fill (DRR-BinFill), are proposed for virtual machine (VM) consolidation. The proposed heuristic algorithms exploit the heterogeneity of the VMs' requirements for different resources by capturing the differences among VMs' demands, and the heterogeneity of the physical machines' resource capacities by capturing the differences among physical machines' resources. The performance of the proposed heuristic algorithms is evaluated with different classes of synthetic workloads under different VM requirement heterogeneity conditions, and the simulation results demonstrate that the proposed heuristics achieve quite similar consolidation performance as dimension-aware heuristics with almost the same computational cost as those of the single dimensional heuristics.

CONGESTION CONTROL, ENERGY EFFICIENCY AND VIRTUAL MACHINE PLACEMENT FOR DATA CENTERS

by
Yan Zhang

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Engineering

Department of Electrical and Computer Engineering

May 2014

APPROVAL PAGE

CONGESTION CONTROL, ENERGY EFFICIENCY AND VIRTUAL MACHINE PLACEMENT FOR DATA CENTERS

Yan Zhang

Dr. Nirwan Ansari, Dissertation Advisor Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Abdallah Khreishah, Committee Member Assistant Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Roberto Rojas-Cessa, Committee Member Associate Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Andrew Sohn, Committee Member Associate Professor of Computer Science, NJIT	Date
Dr. Sotirios Ziavras, Committee Member Associate Provost for Graduate Studies, NJIT Professor of Electrical and Computer Engineering, NJIT	Date

BIOGRAPHICAL SKETCH

Author:	Yan Zhang
Degree:	Doctor of Philosophy
Date:	May 2014

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2014
- Master of Engineering in Communication and Information System,
  Shandong University, Jinan, Shandong, China, 2004
- Bachelor of Engineering in Electronic Information and Technology,
  Shandong University, Jinan, Shandong, China, 2001

Major: Computer Engineering

Presentations and Publications:

Journals:

Y. Zhang and N. Ansari, "Fair Quantized Congestion Notification in Data Center Networks," *IEEE Transactions on Communications*, vol. 61, no.11, pp. 4690-4699, Nov. 2013.

Y. Zhang and N. Ansari, "HERO: Hierarchical Energy Optimization for Data Center Networks," to appear in *IEEE Systems Journal.*

Y. Zhang and N. Ansari, "Multi-sensor Signal Fusion based Modulation Classification by using Wireless Sensor Networks in AWGN Channel," to appear in *Wireless Communications and Mobile Computing*.

Y. Zhang and N. Ansari, "On Protocol-Independent Data Redundancy Elimination," *IEEE Communications Surveys and Tutorial*, vol. 16, no. 1, pp. 455-472, First Quarter 2014.

Y. Zhang and N. Ansari, "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Center Networks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 39-64, First Quarter 2013.

Y. Zhang, N. Ansari, M. Wu, and H. Yu "On Wide Area Network Optimization," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 4, pp. 1090-1113, Fourth Quarter 2012.

Y. Zhang, N. Ansari, and H. Tsunoda, "Wireless Telemedicine Services over Integrated IEEE 802.11/WLAN and IEEE 802.16/WiMAX Networks," *IEEE Wireless Communications*, vol. 17, pp. 30-36, Feb. 2010.

Conferences:

Y. Zhang and N. Ansari, "Heterogeneity Aware Dominant Resource Assistant Heuristics for Virtual Machine Consolidation," in *Proc. of IEEE Global Communications Conference (GLOBCOM)*, Atlanta, GA, USA, Dec. 9-13, 2013.

Y. Zhang, N. Ansari, M. Wu, and H. Yu, "SDRE: Selective Data Redundancy Elimination for Resource Constrained Hosts", in *Proc. of IEEE Global Communications Conference (GLOBCOM)*, Anaheim, California, USA, Dec. 3-7, 2012, pp. 1865-1870.

Y. Zhang, N. Ansari, M. Wu, and H. Yu, "AFStart: An Adaptive Fast TCP Slow Start for Wide Area Networks," in *Proc. of 2012 IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June 10-15, 2012, pp. 1260-1264.

Y. Zhang and N. Ansari, "HERO: Hierarchical Energy Optimization for Data Center Networks," in *Proc. of IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June 10-15, 2012, pp. 2924-2928.

Y. Zhang, N. Ansari, and W. Su "Optimal Decision Fusion Based Automatic Modulation Classification by Using Wireless Sensor Networks in Multipath Fading Channel," in *Proc. of 2011 IEEE Global Communications Conference (GLOBCOM)*, Huston, USA, Dec. 5-9, 2011.

Y. Zhang and N. Ansari, "On Mitigating TCP Incast in Data Center Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Shanghai, P. R. China, April 10-15, 2011, pp. 51-55.

Y. Zhang, N. Ansari, and W. Su, "Multi-sensor Signal Fusion based Modulation Classification by using Wireless Sensor Networks," in *Proc. of 2011 IEEE International Conference on Communications (ICC)*, Kyoto, Japan, June 5-9, 2011.

Patent Disclosures:

N. Ansari and Y. Zhang, "Fair Quantized Congestion Notification (FQCN) to Mitigate Transport Control Protocol (TCP) Throughput Collapse in Data Center Networks," Non-provisional US Patent Application 13/297,101, filed Nov. 15, 2011.

Y. Zhang, N. Ansari, Ming Q. Wu and H. Yu, "System and Method for Transmission Control Protocol Slow Start," (HW 83240743P), Non-provisional US Patent Application Number 13/340,354, filed on 12/29/2011.

N. Ansari and Y. Zhang, “Hierarchical Energy Optimization Algorithm for Data Center Networks,” Non-provisional PCT International Application No. PCT/US12/38891, filed on 5/21/2012.

Y. Zhang, N. Ansari, Ming Q. Wu and H. Yu, “System and Method for Selective Data Redundancy Elimination for Resource-Constrained Hosts,” Non-Provisional US Patent Application Number 13/857,023, filed on April 4, 2013.

N. Ansari and Y. Zhang, “Allocation of Virtual Machines to Physical Machines through Dominant Resource Assisted Heuristics,” Non-provisional PCT International Application No. PCT/US13/72516, filed on 11/29/2013.

To my husband and my son, for their love, sacrifice, patience and support.
To my parents, for their love, support and encouragement.

ACKNOWLEDGMENT

I wish to express my sincere gratitude to my doctoral advisor, Professor Nirwan Ansari, for all his efforts, positive attitude and constructive suggestions that guided me throughout my whole Ph.D. study. His advice, patience and support encouraged me to continuously gain progress in the right direction. Everything that I learned from him was invaluable to me, and without his kind help and guidance, this dissertation could never have been completed.

I also greatly appreciate my committee members, Professor Abdallah Khreishah, Professor Roberto Rojas-Cessa, Professor Andrew Sohn, and Professor Sotirios Ziavras, for their constructive suggestions and careful reviews to improve this dissertation.

This Ph.D. thesis is dedicated to my parents, Dejun Zhang and Guohua Yan, and my husband, Yue Chen, for their understanding, constant support and active accompany throughout this Ph.D study. Additionally, this thesis is also dedicated to my son, Jianlin Chen, who is filling my life with smiles.

I am grateful to my friends from the Advanced Networking Laboratory at NJIT, Tao Han, Mina Taheri Hosseinabadi, Xueqing Huang, Abbas Kiani, Chun-Hao Lo, Haijun Pan, Xiang Sun, Nan Wang, Yi Xu, Si Yi, Chao Zhang, Jingjing Zhang, for interesting discussions and for sharing with me the time, which left me a lot of great memories to last a lifetime.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter Page

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

Data centers, facilities with communications network equipment and servers for data processing and/or storage, are prevalent and essential to provide a myriad of services and applications for various private, non-profit, and government systems. They are used and deployed in nearly every sector of the economy, e.g., financial services, media, universities, and government institutions. Many large data centers have been built in recent years to host online services such as web search and online gaming, distributed file systems such as Google File System (GFS) [1], and distributed Storage System such as BigTable [2] and MapReduce [3]. With rapid deployment of modern high-speed low-latency large-scale data centers, many problems have been observed in data centers, such as data center architecture design, congestion notification, TCP Incast, power consumption, virtual machine placement and load balancing.

1.1 Scope of Study

This thesis focuses on four issues in data center networks: the Ethernet layer or Layer 2 congestion notification, a unified congestion control framework, energy efficiency, and virtual machine placement issues.

1.1.1 Ethernet Layer Congestion Notification in Data Centers

Owing to the inherent merits of Ethernet, such as low cost, ubiquitous connectivity, and ease of management, Ethernet has become the primary network protocol for computer-to-computer communications in a data center. However, Ethernet was designed for best-effort transmissions that may drop packets when the network or switches are busy. In order to address issues raised within a data center, such as increasing demand for higher bandwidth and performance, and low latency interconnect for high performance cluster computing, several congestion notification algorithms have been proposed and developed to reduce or eliminate packet drops at congestion switches in data centers, e.g., Backward Congestion Notification (BCN) [4], Forward Explicit Congestion Notification (FECN) [5], enhanced FECN (E-FECN) [6], and Quantized Congestion Notification (QCN) [7–9]. BCN, FECN, E-FECN and QCN are all queue-based congestion notification algorithms. They all assume that a congestion detector is integrated into each switch, where congestion happens. The switches detect the congestion and generate the feedback congestion messages to the sources based on the calculated congestion measure, and the rate regulators at the sources will adjust the rates of individual flows according to congestion feedback messages received from switches. It has been shown that BCN achieves only proportional fairness but not max-min fairness [10]. FECN and E-FECN can achieve perfect fairness, but the control message overhead is high. The QCN algorithm has been developed for inclusion in the IEEE 802.1Qau standard to provide end-to-end congestion control at the Ethernet Layer or Layer 2 (L2) in DCNs by the IEEE Data Center Bridging Task Group. QCN can effectively control link rates very rapidly in a data center. However, one drawback of QCN is the rate unfairness of different flows when sharing one bottleneck link. One of the main tasks of this thesis is to design an Ethernet layer congestion control algorithm to resolve network congestion efficiently and to improve rate allocation fairness of multiple flows sharing one bottleneck link simultaneously for data center networks.

1.1.2 A Unified Congestion Detection, Notification and Control System for Data Center Networks

Current congestion control algorithms for data center networks implemented on different layers work independently from each other. Every congestion control algorithm requires a congestion detection mechanism, a congestion notification mechanism and a congestion control intelligence method. Such independent congestion control mechanisms may reduce the congestion control efficiency, increase the congestion control overhead, and cannot provide an efficient congestion detection, notification and control over the whole data center network. Primarily, network congestion is detected and notified repeatedly, resulting in the increase of the congestion information messages over the network. In addition, the detected congestion information is not efficiently deployed to resolve network congestion. The congestion control mechanisms deployed on different layers encapsulate the detected congestion information in different congestion notification messages, and therefore, the congestion information detected by congestion control mechanisms cannot be identified and utilized to resolve network congestion uniformly. By capitalizing the architecture of software defined networks (SDNs), network layer congestion control mechanisms can collect the network congestion information over the whole network. However, such global network congestion information is only utilized for rerouting some specific traffic flows to balance the network traffic and is not fully explored for Ethernet layer and transport layer congestion control mechanisms, which can benefit from such global network congestion information. One of the main tasks of this thesis is to design a unified congestion detection, notification and control for data center networks.

1.1.3 Energy Efficiency of Data Centers

In order to provide reliable and scalable computing infrastructure, the high capacity of data centers is especially provisioned for worst-case or busy-hour load, and thus data centers consume a huge amount of energy. And the total cost of ownership (TCO) of physical data center infrastructure, and

It was found that the cost of electrical power consumption contributed to about 20 percent of the total cost of ownership (TCO) of the physical data center infrastructure [11]. However, numerous studies have shown that data center servers rarely operate at full utilization, and it has been well established in the research literature that the average server utilization is often below 30 percent of the maximum utilization in data centers [12], and a great number of servers work in the idle state in these richly-connected data centers. At low levels of workload, servers are highly energy-inefficient. The high operational costs and the mismatch between the data center utilization and power consumption have spurred great interest in improving data center energy efficiency. Therefore, another task of this thesis is to optimize the power consumptions and improve the energy efficiency of data centers.

1.1.4 Virtual Machine Placement

Machine virtualization is one of the mostly used techniques to provide flexible and cost-effective resource sharing among users in data centers. Consolidating virtual machines (VMs) to physical servers in data center infrastructure helps to increase the utilization of data center resources, improve the energy efficiency of data centers since the idle infrastructures can be shut down or put into the low power mode to save energy, balance the workload across the whole data center to avoid hotspots, and ensure the quality of service for cloud computing applications. In general, VM consolidation is formulated as a vector bin packing problem, which is a well-known NP-hard problem. Hence, heuristic algorithms, such as single dimensional heuristics (e.g., first fit decreasing (FFD)) and dimension-aware heuristics (e.g., DotProduct), are usually deployed in practice for VM consolidation. However, all of these previous heuristic algorithms do not sufficiently explore the heterogeneity of the VMs' resource requirements and the heterogeneity of the physical servers' resource capacities. VM consolidation by exploring the heterogeneity among VMs and physical servers is another task of this thesis.

1.2 Organization

The rest of this thesis is organized as follows. Chapter 2 presents an enhanced QCN congestion notification algorithm, called fair QCN (FQCN), to improve rate allocation fairness of multiple flows sharing one bottleneck link in DCNs. Chapter 3 describes a unified congestion detection, notification and control framework to efficiently reduce or eliminate network congestion and to simultaneously ensure fairness among the sharing traffic flows for data center networks. Chapter 4 shows a two-level power optimization model, namely, Hierarchical EneRgy Optimization (HERO), to reduce the power consumption of DCNs by switching off network switches and links while still guaranteeing full connectivity and maximizing link utilization. Chapter 5 describes several heterogeneity aware dominant resource assistant heuristic algorithms for VM consolidation. Finally, Chapter 6 discusses some future research directions, and concludes the thesis.

CHAPTER 2

FAIR QUANTIZED CONGESTION NOTIFICATION IN DATA CENTERS

Quantized Congestion Notification (QCN) has been developed for IEEE 802.1Qau to provide congestion control at the Ethernet Layer or Layer 2 in data center networks (DCNs) by the IEEE Data Center Bridging Task Group. One drawback of QCN is the rate unfairness of different flows when sharing one bottleneck link. In this chapter, an enhanced QCN congestion notification algorithm, called fair QCN (FQCN), is proposed to improve rate allocation fairness of multiple flows sharing one bottleneck link in data center networks. FQCN identifies congestion culprits through joint queue and per flow monitoring, feedbacks individual congestion information to each identified congestion culprit through multi-casting, and ensures convergence to statistical fairness. The stability and fairness of FQCN is analyzed by using Lyapunov functions and the performance of FQCN is evaluated through simulations in terms of the queue length stability, link throughput and rate allocations to traffic flows with different traffic dynamics under three network topologies, namely, the dumb-bell topology, parking-lot topology, and a simple and representative TCP Incast network topology. Simulation results confirm the rate allocation unfairness of QCN, and validate that FQCN can maintain the queue length stability, successfully allocate the fair share rate to each traffic source sharing the link capacity, and enhance TCP throughput performance in the TCP incast setting.

2.1 Introduction

Owing to the inherent merits of Ethernet, such as low cost, ubiquitous connectivity, and ease of management, Ethernet has become the primary network protocol to provide a consolidated network solution for data center networks (DCNs). However, Ethernet was originally designed for best-effort communications in a local area network (LAN) and not optimized for data center networks. The Data Center Bridging Task Group [13] in the IEEE 802.1 Ethernet standards body thus aims to enhance classical switched Ethernet to provide more services for DCNs. Project IEEE 802.1Qau is concerned with specifications of the Ethernet Layer or Layer 2 congestion notification mechanism for data center networks.

Several congestion notification algorithms have been proposed and developed to reduce or eliminate packet drops at the congestion switch in DCNs, e.g., Backward Congestion Notification (BCN) [4], Forward Explicit Congestion Notification (FECN) [5], enhanced FECN (E-FECN) [6], and Quantized Congestion Notification (QCN) [7–9]. It has been shown that BCN achieves only proportional fairness but not max-min fairness [10]. FECN and E-FECN can achieve perfect fairness, but the control message overhead is high. QCN can effectively control link rates to resolve switch congestion in several round trips, i.e., several hundred micro seconds with 10 Gbps link capacity in today's typical DCNs. However, one drawback of QCN is the rate unfairness of different flows when sharing one bottleneck link. Such rate unfairness also degrades the TCP throughput in synchronized readings of data blocks across multiple servers [14]. Thus, an enhanced QCN congestion control algorithm, called fair Quantized Congestion Notification (FQCN) [15, 16], is proposed to improve fairness of multiple flows sharing one bottleneck link. The main idea of FQCN is to distinguish congestion culprits through queue and per flow monitoring and feedback the global and per-flow congestion information to all culprits through multi-casting when the switch is congested.

The rest of the chapter is organized as follows. In Section 2.2, related work are summarized. The QCN algorithm is first introduced in Section 2.3. Section 2.4 details the proposed FQCN congestion notification algorithm and analyzes the performance of FQCN by using the fluid model and Lyapunov method. Then, the performance of the proposed FQCN is evaluated in Section 2.6. Section 2.7 concludes the Chapter.

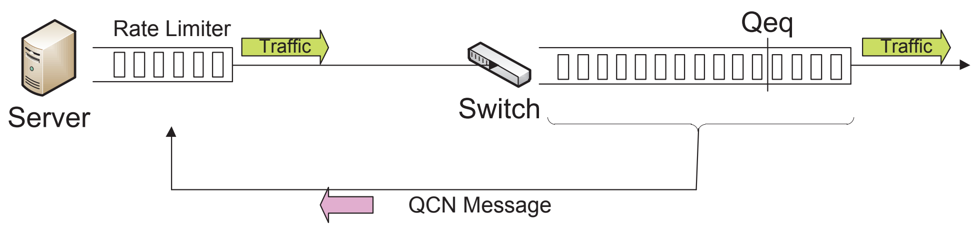
2.2 Related Work

BCN [4], FECN [5], E-FECN [6] and QCN [7–9] are all queue sampling based congestion notification algorithms. They all assume that the switches detect congestion by sampling the queue and generate the feedback congestion notification messages to the sources based on the calculated congestion parameter. The rate regulators at the sources adjust the rates of individual flows according to congestion notification messages received from the switches.

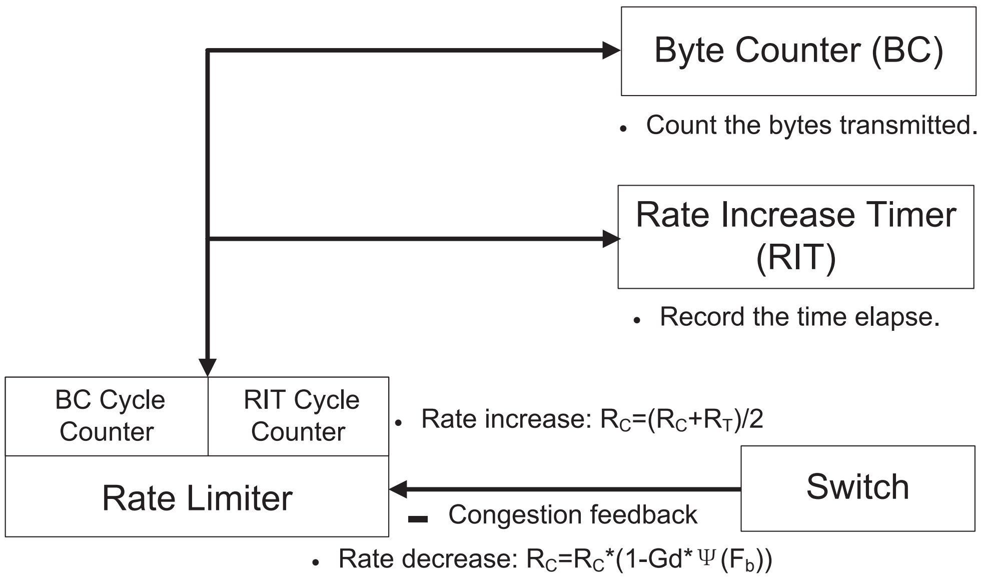
Both BCN [4] and QCN [7–9] are rate-based closed-loop feedback control mechanisms between the rate regulator located at a source and a switch. In BCN, a switch feeds back not only negative information regarding congestion but also positive rate increasing information, and the rate regulator in BCN adjusts its rate by using a modified Additive Increase and Multiplicative Decrease algorithm. In QCN, only negative congestion feedback is used. Thus, the control overhead of QCN is smaller than that of BCN. Since there is no positive feedback in QCN, the rate regulators in QCN provide mechanisms to recover the lost bandwidth and probe for extra available bandwidth voluntarily. In addition, the sampling probability is constant in BCN while it is proportional to the congestion parameter in QCN.

To ensure scalability, one of the main requirements and design rationales of IEEE 802.1 for QCN is to enforce the QCN enabled switch to be "no state", i.e., not storing any per flow state. This requirement also prevents QCN from providing any fairness stronger than proportional fairness, such as max-min fairness. An enhancement to QCN, called approximate fairness QCN (AF-QCN) [17], was proposed to improve the fairness allocation of link capacity among all sharing sources. AF-QCN extends the functionality of QCN with an approximate fairness controller to provide weighted fairness on a per-flow or per-class basis. For each incoming flow, the Ethernet switch with AF-QCN estimates its arrival rate, calculates its fair share, and derives its dropping probability.

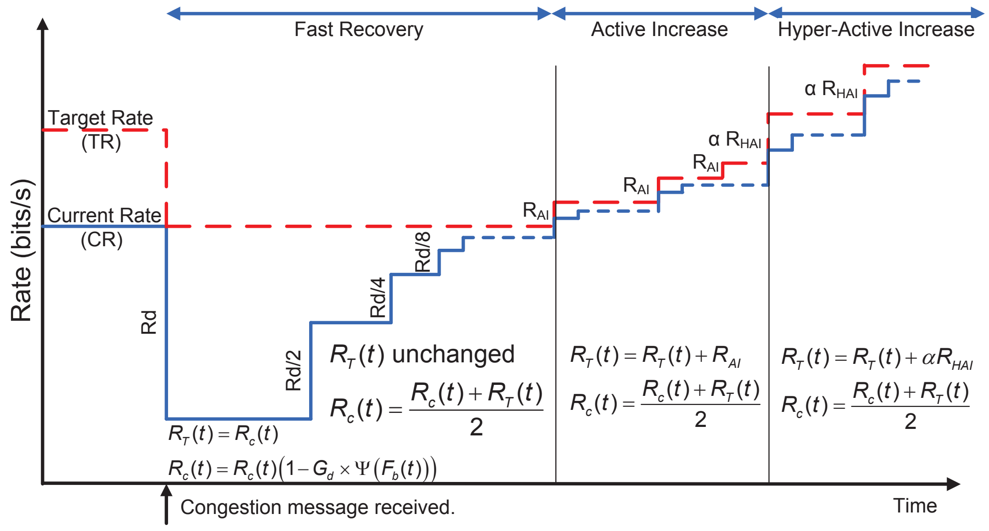
FECN [5] is a close-loop end-to-end explicit rate feedback control mechanism. Each source periodically probes the congestion condition along the path to the destination. The rate field in the probe message is modified along the forwarding path by each switch if the available bandwidth at the switch in the forwarding path is smaller than the value of the rate field in the probe message. When a source receives the probe message returned back from the destination, the rate regulator adjusts the sending rate as indicated in the received message. All flows are treated fairly in FECN since the same rate is advertised by the



(a) QCN system model.


Byte Counter (BC)
• Count the bytes transmitted.
Rate Increase Timer (RIT)
• Record the time elapse.
BC Cycle Counter
RIT Cycle Counter
• Rate increase: $R_C=(R_C+R_T)/2$
Rate Limiter
Switch
Congestion feedback
• Rate decrease: $R_C=R_C*(1-Gd*\Psi(F_b))$

(b) QCN control mechanism.



(c) QCN rate limiter operation.

Figure 2.1 QCN overview.

switch. The source rate in BCN and QCN can be reduced more quickly than that in FECN because the message in BCN and QCN is sent directly from the congestion point while the probe message in FECN has to take a round trip before it returns back to the source. Also, the congestion control message overhead of FECN is higher than that of QCN, because the probe message in FECN travels periodically a round trip from the source to the destination while the congestion control message in QCN is generated only when the switch congestion occurs and transmitted through a much shorter distance from the congested switch to the source. E-FECN [6] enhances FECN by allowing the switches feedback to the source directly under severe congestion. Reference [18] provides a more detailed discussion on the Ethernet layer congestion notification algorithms for data center networks.

2.3 Quantized Congestion Notification

The QCN algorithm is composed of two parts as shown in Figure 2.1(a): switch or congestion point (CP) dynamics and rate limiter (RL) dynamics. At CP, the switch buffer attached to an oversubscribed link samples incoming packets and feeds the congestion severity level back to the source of the sampled packet. At one traffic source, RL decreases its sending rate based on the congestion notification message received from CP, and increases its rate voluntarily to recover lost bandwidth and probe for extra available bandwidth. The rate control mechanism and the operations of RL in QCN are summarized in Figure 2.1(b) and (c), respectively. In order to ease the understanding, a list of all symbols used in this chapter is summarized in Table 2.1.

2.3.1 The CP Algorithm

The goal of CP is to maintain the buffer occupancy at a desired operating point, Q_{eq}, which is the queue threshold for equilibrium. At time t, CP samples the incoming packets with a sampling probability $p(t)$, and computes the congestion feedback value $F_b(t)$. The sampling probability is initialized to 1%, and is updated after computing the congestion feedback value $F_b(t)$ at each sampling event. Denote $Q_{len}(t)$ and $Q_{len}(t - \tau)$ as the

Table 2.1 Description of Symbols

Symbols	**Description**
	Network and Traffic Parameters
$\mathcal{L}$	The set of links.
C_l	The capacity of link l.
$\mathcal{S}$	The set of sources.
$\mathcal{S}_l$	The set of sources using link l.
$\mathcal{S}_l^L$	The set of low rate flows on link l.
$\mathcal{S}_l^H$	The set of high rate flows on link l.
$\mathcal{S}_l^R$	The set of overrated flows on link l.
$R_i(t)$	The rate of source i.
$\overrightarrow{R(t)}$	The vector of rates of all sources.
$a_{l,i}$	The fraction of traffic from source i over the link l.
$f_l(t)$	The total amount of traffic over link l.
	Queue Parameters
$Q_{len}(t)$	The instantaneous queue length at time t.
Q_{eq}	The queue threshold for equilibrium.
$Q_{over}(t)$	The excess queue length that exceeds the equilibrium Q_{eq} at time t.
Q_δ(t)	The queue variation over the last sampling interval.
	QCN Parameters
$F_b(t)$	The congestion feedback value calculated at time t.
$\Psi(F_b(t))$	The quantized congestion feedback value of $F_b(t)$.
w	The weight parameter for the queue growth rate $Q_\delta(t)$ in computing the congestion feedback value $F_b(t)$.
G_d	The multiplicative decrease parameter of rate.
$p(t)$	The time varying sampling probability with which the CP samples the incoming packets.
C_T	The cycle threshold to determine the state of byte counter or rate increase timer, in the FR or AI phase.
B_L	Bytes transmitted to complete one byte counter cycle.
T	The Rate Increase Timer period in milliseconds.
$R_C(t)$	The current sending rate.
$R_T(t)$	The target sending rate
R_{AI}	The sending rate incremental parameter when one cycle of BC or RIT completes in AI phase.
R_{HAI}	The sending rate incremental parameter when one cycle of BC or RIT completes in HAI phase.
	FQCN Parameters
$B_i(t)$	The byte counts of flow i at time t.
W_i	The weight coefficient for flow i.
$M_i(t)$	The fair share for flow i on link l at the sampling time t.
$M_i^F(t)$	The fine-grained fair share for high rate flow $i \in \mathcal{S}_l^H$ on link l at the sampling time t.
$\Psi_{F_b}^F(i,t)$	The quantized congestion feedback value calculated with FQCN for the overrated source i at time t.
	Analysis Parameters
$p_l(f_l(t))$	The probability of generating a QCN congestion message from link l with load $f_l(t)$ at time t.
$P_l(t)$	The cumulative probability of generating a QCN congestion message from link l at time t.
$P(\overrightarrow{R(t)})$	The probability of generating a QCN message from all links at time t with vector of rates $\overrightarrow{R(t)}$.
$J_A(\overrightarrow{R})$	The Lyapunov function of vector $\overrightarrow{R}$.

To ease the reading and analysis, bits and bits/s are used as the units of queue length and transmission rate, respectively.

instantaneous queue length in bits of the current sampling event at time t and last sampling event at time $t - \tau$, respectively, where τ is the time interval between two adjacent sampling events. The congestion feedback value $F_b(t)$ consists of a weighted sum of the instantaneous queue offset $Q_{over}(t) = Q_{len}(t) - Q_{eq}$ and the queue variation over the last sampling interval $Q_\delta(t) = Q_{len}(t) - Q_{len}(t - \tau)$, as defined below:

$$F_b(t) = -(Q_{over}(t) + w * Q_\delta(t)) \tag{2.1}$$

where w is a non-negative constant, taken to be 2 for the baseline implementation. In fact, $Q_{over}(t)$ indicates queue-size excess while $Q_\delta(t)$ indicates rate excess. If $F_b(t)$ is negative, either the buffer or the link or both are oversubscribed and a congestion notification message containing the value of quantized $|F_b(t)|$, denoted as $\Psi(F_b(t))$, is sent back to the source of the sampled packet; otherwise, no feedback message is sent.

At each sampling event, the sampling probability is updated as follows:

$$p(t) = \begin{cases} (1 + \frac{9}{64}\Psi(F_b(t)))\% & (F_b(t) < 0) \\ 1\% & (F_b(t) \geq 0) \end{cases} \tag{2.2}$$

From the above equation, it can be observed that if there is no congestion ($F_b(t) \geq 0$), CP checks the congestion status with a probability of 1%; otherwise, the sampling probability is increased as a linear function of the quantized congestion feedback value $\Psi(F_b(t))$. In the default implementation, the congestion feedback value $F_b(t)$ is quantized to six bits and the maximum quantized value of $F_b(t)$ is 64, and therefore, the maximum sampling probability is 10%.

2.3.2 The RL Algorithm

As shown in Figure 2.1 (c), RL adjusts the sending rate of the associated traffic source by decreasing the sending rate based on the quantized congestion feedback value contained in the congestion notification message, and increasing the sending rate voluntarily to recover lost bandwidth and probe for extra available bandwidth.

Rate Decrease When a congestion notification message is received, the current sending rate $R_C(t)$ is set as the target rate $R_T(t)$ and the current rate is reduced by a factor of $R_d(t) = R_C(t)G_d \times \Psi(F_b(t))$ as follows:

$$\begin{aligned} R_T(t) &= R_C(t) \\ R_C(t) &= R_C(t)(1 - G_d \times \Psi(F_b(t))) \end{aligned} \tag{2.3}$$

where the constant G_d is chosen to ensure that the sending rate cannot decrease by more than 50%, and thus $G_d * \Psi(F_{bmax}) = \frac{1}{2}$, where F_{bmax} denotes the maximum of $F_b(t)$. In the default implementation, the maximum quantized value of $F_b(t)$ is 64; thus, G_d is configured to 1/128.

Rate Increase Two modules, Byte Counter (BC) and Rate Increase Timer (RIT), are introduced in RL for rate increases. BC is a counter for counting the number of bytes transmitted by the traffic source, which is used to increase the sending rate by RL. Based on BC only, it will take a long time for a low rate source to increase its sending rate; this might result in low bandwidth utilization if there is tremendous available bandwidth. Besides, the unfair sharing of bottleneck bandwidth can be observed when a low rate flow competes for the bandwidth with a high rate flow. In order to enable fast bandwidth recovery, rather than getting stuck at a low sending rate for a long time, RIT is introduced in RL to increase the source's sending rate periodically instead of based on the amount of data transferred as BC does.

BC and RIT work in two phases, Fast Recover (FR) and Active Increase (AI). At the FR phase, BC counts data bytes transmitted and increases the BC cycle by 1 when B_L bytes are transmitted. After each cycle, RL increases its rate to recover some of the bandwidth it lost at the previous rate decrease episode. After the C_T cycle (where C_T is a constant chosen to be 5 cycles in the baseline implementation), BC enters the AI phase to probe for extra bandwidth on the path. In the AI phase, RL counts the data bytes transmitted and increases the BC cycle by 1 when $B_L/2$ bytes are transmitted. BC is reset every time a rate decrease is applied and enters the FR state. RIT functions similarly as BC. In the FR phase, RIT completes one cycle with $T\ ms$ duration. After it counts out C_T cycles of $T\ ms$ duration, it enters the AI phase where each cycle is set to $T/2\ ms$ long. It is reset when a congestion notification message arrives and enters the FR phase.

RL works in the FR, AI or Hyper-Active Increase (HAI) phase depending on the state of BC and RIT. BC and RIT jointly determine rate increases of RL, when either BC or RIT completes one cycle of data transfer, which will trigger a rate increase event. When a rate increase event occurs, the update of the current rate $R_C(t)$ and target rate $R_T(t)$ in different RL state is summarized as follows:

a) RL is in FR if both BC and RIT are in FR. In this case, when either BC or RIT completes a cycle, the target rate $R_T(t)$ remains unchanged while the current rate $R_C(t)$ is updated as:

$$R_C(t) = \frac{1}{2}(R_C(t) + R_T(t)) = R_c(t) + \frac{1}{2}(R_T(t) - R_c(t)) \tag{2.4}$$

b) RL is in AI if either BC or RIT is in AI. In this case, when either BC or RIT completes a cycle, the current rate and target rate are updated as:

$$\begin{aligned} R_T(t) &= R_T(t) + R_{AI} \\ R_C(t) &= \tfrac{1}{2}(R_C(t) + R_T(t)) \end{aligned} \tag{2.5}$$

where R_{AI} is a constant sending rate increment for RL in the AI state, and it is set to be 5 Mbps in the baseline implementation.

c) RL is in HAI if both BC and RIT are in AI. In this case, the target rate and current rate are updated as:

$$\begin{aligned} R_T(t) &= R_T(t) + \alpha * R_{HAI} \\ R_C(t) &= \tfrac{1}{2}(R_C(t) + R_T(t)) \end{aligned} \tag{2.6}$$

where R_{HAI} is a constant sending rate increment for RL in the HAI state, and is set to 50 Mbps in the baseline implementation, and α is the minimum cycle count of BC and RIT in the AI state.

2.4 FQCN

One drawback of QCN is the rate allocation unfairness of different flows when sharing one bottleneck link, which can be seen clearly in our simulation results shown in Section 2.6. To improve the fair allocation of link capacity among all sharing sources in QCN, the proposed fair Quantized Congestion Notification (FQCN) identifies congestion culprits, which send packets at rates above their fair shares of the bottleneck link capacity, and feeds the congestion notification messages back to all these identified congestion culprits. FQCN differs from QCN at the CP algorithm. As shown in Eq. (2.1), QCN randomly samples the traffic packets at CP and feedbacks a global congestion status sequentially to a single traffic source deemed as the congestion culprit. This random congestion feedback in QCN prevents the sending rates from converging to statistical fairness. FQCN addresses these deficiencies: 1) it identifies the overrated flows, which are deemed as congestion culprits and whose sending rates are larger than their fair share rates; 2) through joint queue and per flow monitoring, it feedbacks individual congestion status to each culprit through multi-casting, thus ensuring convergence to statistical fairness.

2.4.1 Congestion Culprits Identification

Consider a network consisting of a set $\mathcal{L} = \{1, \cdots, L\}$ of links of capacity C_l ($l \in \mathcal{L}$). The network is shared by a set $\mathcal{S} = \{1, \cdots, S\}$ of sources. $\overrightarrow{R(t)} = [R_1(t), \cdots, R_S(t)]$ is the vector of rates of all sources with $R_i(t)$ denoting source i's sending rate. $\mathcal{S}_l$ is the set of sources using link l.

In each sampling period, which is the time interval between two adjacent packet sampling events, the switch monitors the queue length, the number of arriving and departing packets to calculate the congestion feedback value using Eq. (2.1) as in QCN. The switch also monitors the number of total bytes received for each flow $B_i(t)$ ($i \in \mathcal{S}_l$), which shares one bottleneck link l within one sampling interval at time t. The switch identifies congestion culprits by using a two-step approach, which removes the influence of low rate flows to enable precise identification of congestion culprits.

Step 1: Identify high rate flows. The fair share for each flow $i \in \mathcal{S}_l$ on link l at the sampling time t can be estimated as:

$$M_i(t) \quad = \quad \tfrac{W_i}{\sum_{i \in \mathcal{S}_l} W_i} \textstyle\sum_{i \in \mathcal{S}_l} B_i(t) \tag{2.7}$$

where W_i is the weight coefficient for flow i, which can be determined by traffic class, source address, destination address, etc. Thus, the traffic flows can be classified into two categories by comparing $B_i(t)$ with $M_i(t)$ ($i \in \mathcal{S}_l$). A traffic flow i from which the total number of bytes received at the switch $B_i(t)$ is smaller than its estimated fair share $M_i(t)$ ($B_i(t) < M_i(t)$) will be assigned to the low rate source set $\mathcal{S}_l^L = \{\forall i \in \mathcal{S}_l | B_i(t) < M_i(t)\}$. Otherwise, it will be assigned to the high rate source set $\mathcal{S}_l^H = \{\forall i \in \mathcal{S}_l | B_i(t) \geq M_i(t)\}$.

Step 2: Identify congestion culprits in the high rate flow set $\mathcal{S}_l^H$. The fair share can be fine-grained among the high rate source set $\mathcal{S}_l^H$ as:

$$M_i^F(t) \quad = \quad \tfrac{W_i}{\sum_{i \in \mathcal{S}_l^H} W_i} \textstyle\sum_{i \in \mathcal{S}_l^H} B_i(t) \tag{2.8}$$

Similar to the identification of high rate flows, the congestion culprits can be identified by comparing $B_i(t)$ with $M_i^F(t)$ ($i \in \mathcal{S}_l^H$). The source in the high rate source set $\mathcal{S}_l^H$ with the total number of bytes received at the switch equal to or larger than its fine-grained fair share $M_i^F(t)$ is considered as an overrated flow, which is deemed as a congestion culprit. All of overrated flows form an overrated flow set $\mathcal{S}_l^R = \{i \in \mathcal{S}_l^H | B_i(t) \geq M_i^F(t)\}$.

2.4.2 Per Flow Congestion Parameter Calculation

If the calculated congestion feedback value $F_b(t)$ using Eq. (2.1) is negative, the congestion notification message will be sent back to all identified overrated flows through multi-casting. For each overrated flow $i \in \mathcal{S}_l^R$, the quantized congestion feedback value $\Psi_{F_b}(i,t)$ in the congestion notification message to the source of the overrated flow $i \in \mathcal{S}_l^R$ is calculated as follows:

$$\Psi_{F_b}(i,t) \quad = \quad \tfrac{B_i(t)/W_i}{\sum_{k \in \mathcal{S}_l^R} B_k(t)/W_k} \times \Psi(F_b(t)) \tag{2.9}$$

From the above equation, it can be observed that the quantized congestion feedback value $\Psi_{F_b}(i,t)$ in each congestion notification message is proportional to $\Psi(F_b(t))$ and the total number of bytes received at the switch normalized by its weight coefficient $B_i(t)/W_i$, while the sum of the congestion feedback values for all congestion culprits is equal to $\Psi(F_b(t))$. Also, $\Psi_{F_b}(i,t)$ is quantized to 6 bits because $F_b(t)$ is quantized to six bits.

Figure 2.2 describes the FQCN system model with three source flows. In this example, sources 1 and 3 are identified as congestion culprits, and the congestion notification message with the quantized congestion feedback values $\Psi_{F_b}(1)$ and $\Psi_{F_b}(3)$, calculated according to Eq. (2.9), are fed back to sources 1 and 3, respectively.

FQCN differs from QCN as well as AF-QCN in computing the congestion feedback value. QCN does not distinguish flow-dependent congestion information and feedbacks the same congestion status to the source of the randomly sampled packet, while the congestion feedback value in the congestion notification message in both AF-QCN and FQCN is flow-dependent. FQCN distinguishes from QCN as well as AF-QCN by sending QCN congestion notification message to the sources of all congestion culprits rather than to the source of the randomly sampled packet in QCN and AF-QCN. Thus, the switch congestion is resolved much faster with FQCN than that with QCN or AF-QCN. Furthermore, the signaling overhead of FQCN is lighter than that of QCN and AF-QCN because congestion messages could also be received by low rate sources, which are not the real congestion culprits. That is, throttling low rate sources are not as effective as throttling high rate sources in mitigating congestion. AF-QCN increases the congestion control algorithm complexity with arrival rate estimation, fair share calculation, and dropping probability calculation for each flow. AF-QCN can also identify congestion culprits, but it still feeds congestion information only back to the source of the randomly sampled packet.

2.5 Analysis of FQCN

The stability and fairness of FQCN can be analyzed via the Lyapunov method [19], which is an important tool to determine the stability of the equilibrium solution of a system

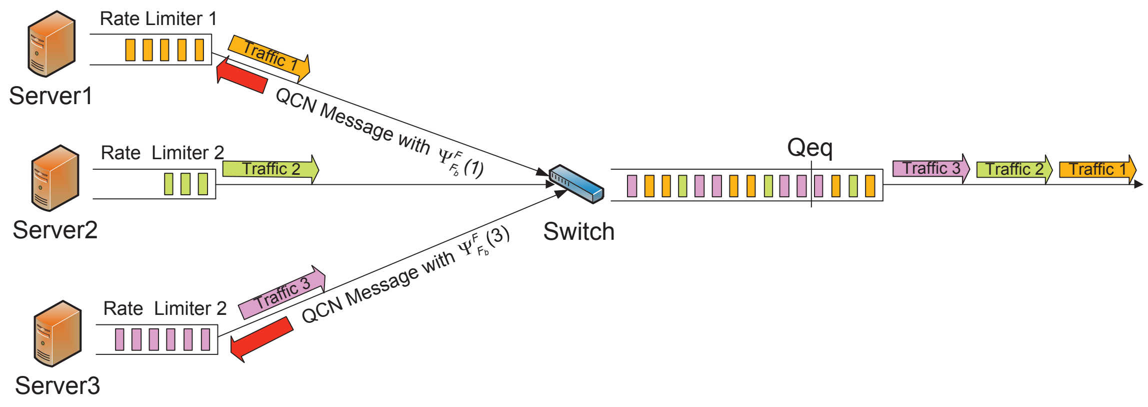


Figure 2.2 FQCN overview.

of ordinary differential equations (ODEs). In this section, a simplified FQCN with RIT disabled was analyzed.

As introduced in Section 2.4, the FQCN control mechanism can be separated into two parts: CP control and RL control. At the switch, by assuming that the queue length is differentiable, the switch dynamics are given by:

$$\frac{dQ_{len}(t)}{dt} = \sum_{i \in \mathcal{S}_l} R_i(t) - C_l \tag{2.10}$$

Based on Eq. (2.1), the congestion feedback value $F_b(t)$ generated by link l can be calculated by:

$$F_b(t) \quad = \quad -(Q_{len}(t) - Q_{eq}) - \tfrac{w}{C_l * p(t)}(\sum_{i \in \mathcal{S}_l} R_i(t) - C_l) \tag{2.11}$$

where $p(t)$ is the time-varying sampling probability at the switch. This equation is the continuous version of calculating the congestion feedback value of Eq. (2.1). If $F_b(t)$ is negative, each congestion notification message is sent back to individual sources of the overrated flows with congestion parameter calculated by Eq. (2.9).

RP adjusts the sending rate by performing additive increase and multiplicative decrease (AIMD). At each rate increase event, the sending rate is increased by $(R_i^T(t) - R_i(t))/2$, and the rate increase interval is $8B_L/R_i(t)$ with BC, where $R_i^T(t)$ and $R_i(t)$ are the target and current sending rate of source i, respectively. Therefore, the expected rate increase per unit time is $\frac{(R_i^T(t)-R_i(t))R_i(t)}{16B_L}$.

Denote $p_l(f_l(t))$ as the probability that a congestion message will be sent to each overrated source $i \in S_l^R$ at each packet sampling event when the load on link l is $f_l(t)$. This probability is a monotonically increasing and differentiable function of the load [20], $f_l(t) = \sum_{j\in\mathcal{S}_l} a_{l,j}R_j(t)$, where $a_{l,j} = \{0,1\}$ indicates whether the traffic from source j flows on link l ($a_{l,j} = 1$) or not ($a_{l,j} = 0$). Note that the splitting of traffic over multiple parallel paths is not considered because the split traffic flows between the same source and destination pair can be treated as multiple flows independently. When a negative congestion message is received by RP at source i, it will reduce the sending rate by $G_d\Psi_{F_b}(i,t)R_i(t)$. The expected interval between successive congestion messages received at an overrated source is equal to $K/(C_l\sum_{l\in\mathcal{L}} p_l(f_l(t))$, where K is the packet size in bits. Therefore, the expected rate decrease per unit time is $G_d\Psi_{F_b}(i)C_l\sum_{l\in\mathcal{L}} p_l(f_l(t)R_i(t)/K$, where $\Psi_{F_b}(i)$ is the expected congestion feedback value received at source i.

An ODE describing the expected rate change at source i can be expressed as:

$$\frac{dR_i(t)}{dt} = \frac{(R_i^T(t)-R_i(t))R_i(t)}{16B_L} \times (1-\sum_{l\in\mathcal{L}} p_l(f_l(t))) - \frac{G_d\Psi_{F_b}(i)C_lR_i(t)}{K} \times \sum_{l\in\mathcal{L}} p_l(f_l(t)) \tag{2.12}$$

In order to study the attractors of the ODE Eq. (2.12), a Lyapunov function for this ODE [19–21] was identified as follows:

$$\sum_{l\in\mathcal{L}} p_l(f_l(t)) = \frac{\partial}{\partial R_i(t)}\sum_{l\in\mathcal{L}} P_l(f_l(t)) = \frac{\partial P\big(\overrightarrow{R(t)}\big)}{\partial R_i(t)} \tag{2.13}$$

where $P_l(t)$ is a primitive of $p_l(\overrightarrow{R_l(t)})$ defined by

$$P_l\big(\overrightarrow{R_l(t)}\big) = \int_0^{f_l(t)} p_l(u)\mathrm{d}u$$

and

$$P\left(\overrightarrow{R(t)}\right) = \sum_{l\in\mathcal{L}} P_l\left(f_l(t)\right)$$

Here, $P_l(f_l(t))$ is the cumulative probability of generating a negative QCN congestion notification message from link l with the link load $f_l(t)$, and $P(\overrightarrow{R(t)})$ is the cumulative probability of generating a congestion notification message from all links with the vector of rates $\overrightarrow{R(t)}$.

The ODE Eq. (2.12) can be rewritten as:

$$\frac{dR_i(t)}{dt} = \frac{R_i(t)[\alpha_i - KR_i(t)]}{16KB_L} \times \left\{ \frac{K(R_i^T(t) - R_i(t))}{\alpha_i - KR_i(t)} - \frac{\partial P(\overrightarrow{R(t)})}{\partial R_i(t)} \right\} \tag{2.14}$$

where $\alpha_i = KR_i^T + 16B_L C_l G_d \Psi_{F_b}(i)$.

The Lyapunov function for the ODE Eq. (2.14) is identified as:

$$J_A(\vec{R}) = \sum_{i \in \mathcal{S}} \Phi(R_i) - P(\vec{R}) \tag{2.15}$$

with

$$\Phi(R_i) = \int_0^{R_i} \frac{K(R_i^T - v_i)}{\alpha_i - KR_i(t)} \mathrm{d}v_i = R_i + \frac{16B_L C_l G_d \Psi_{F_b}(i)}{K} \log{(1 - \frac{K}{\alpha_i} R_i)} \tag{2.16}$$

The Lyapunov function Eq. (2.15) for the ODE Eq. (2.14) can be transformed to:

$$J_A(\vec{R}) = \sum_{i \in \mathcal{S}} R_i + \sum_{i \in \mathcal{S}} \frac{16B_L C_l G_d \Psi_{F_b}(i)}{K} \log{(1 - \frac{K}{\alpha_i} R_i)} - P(\vec{R}) \tag{2.17}$$

Similarly, following the same above procedure by replacing the expected interval between successive congestion messages received at source i with $K/(R_i(t) \sum_{l \in \mathcal{L}} p_l(f_l(t)))$, a Lyapunov function $J_B(\vec{R})$ can also be identified for a QCN system as:

$$J_B(\vec{R}) = \frac{K}{K - \eta} \sum_{i \in \mathcal{S}} [R_i + \frac{\eta R_i^T}{K - \eta} \log{(1 - \frac{K - \eta}{KR_i^T} R_i)}] - P(\vec{R}) \tag{2.18}$$

with $\eta = 16B_L G_d \Psi_{F_b}$, where Ψ_{F_b} is the expected congestion feedback value.

2.5.1 Stability Analysis

Proposition 1: The strictly concave function $J_A(\vec{R})$ is a Lyapunov function for the differential equation of the FQCN system Eq. (2.14), which is stable, and the unique value

$\overrightarrow{R}$ that maximizes $J_A(\overrightarrow{R})$ is an equilibrium point of the system, to which all trajectories converge and maximize the link utilization.

Proof: From Eq. (2.17), it is easy to see that $J_A(\overrightarrow{R})$ is strictly concave over the bounded region ($\overrightarrow{R} \succeq 0$) with an interior maximum, and therefore the value of $\overrightarrow{R}$ that maximizes $J_A(\overrightarrow{R})$ is unique. The equilibrium rates $\overrightarrow{R}$ that maximizes $J_A(\overrightarrow{R})$ can be identified by setting the following derivatives to zero:

$$\frac{\partial J_A(\overrightarrow{R(t)})}{\partial R_i(t)} = \frac{K(R_i^T(t) - R_i(t))}{\alpha_i - KR_i(t)} - \frac{\partial P(\overrightarrow{R(t)})}{\partial R_i(t)} \tag{2.19}$$

Further, along any solution of $\overrightarrow{R(t)}$, its gradient over t is:

$$\begin{aligned}\frac{d}{dt}J_A(\overrightarrow{R(t)}) &= \sum_{i\in\mathcal{S}} \frac{\partial J_A(\overrightarrow{R(t)})}{\partial R_i(t)}\frac{dR_i(t)}{dt} \\ &= \sum_{i\in\mathcal{S}} \frac{R_i(t)[\alpha_i - KR_i(t)]}{16KB_L}\left(\frac{\partial J_A(\overrightarrow{R(t)})}{\partial R_i(t)}\right)^2\end{aligned} \tag{2.20}$$

Thus, $J_A(\overrightarrow{R(t)})$ is strictly increasing with t, unless $\overrightarrow{R(t)} = \overrightarrow{R}$, the unique $\overrightarrow{R}$ that maximizes $J_A(\overrightarrow{R})$. The function $J_A(\overrightarrow{R})$ is thus a Lyapunov function for the differential equations of the FQCN system Eq. (2.14). The maximization of $J_A(\overrightarrow{R})$ is constrained by the link capacity, and hence the equilibrium rates $\overrightarrow{R}$ maximize the link utilization. Since the equilibrium rates are stable, the queue is also stable. Similar results can be observed with the Lyapunov function $J_B(\overrightarrow{R})$ for the QCN system. Hence, both FQCN and QCN are stable, and the rate of each source converges to an equilibrium point that maximizes the link utilization.

2.5.2 Fairness Analysis of FQCN

Proposition 2: With multiple sources sharing a link, FQCN is closer to max-min fairness than to proportional fairness.

Proof: As shown in [20], the expectation of the congestion feedback $p_l(f_l)$ at link l is close to a Dirac function in the limiting case as:

$$\delta_{C_l}(f_l) = \begin{cases} 0 & \text{if } (f_l < C_l) \\ +\infty & \text{if } (f_l \geq C_l) \end{cases}$$

Thus, following the method in [20], the rates with FQCN are distributed so as to maximize:

$$F_A(\vec{R}) = \sum_{i \in \mathcal{S}} \frac{16 B_L C_l G_d \Psi_{F_b}(i)}{K} \log{(1 - \frac{K}{\alpha_i} R_i)} \tag{2.21}$$

subject to the constraints:

$$\sum_{l \in \mathcal{L}} a_{l,i} R_i \leq C_l, \forall l \in \mathcal{L}$$

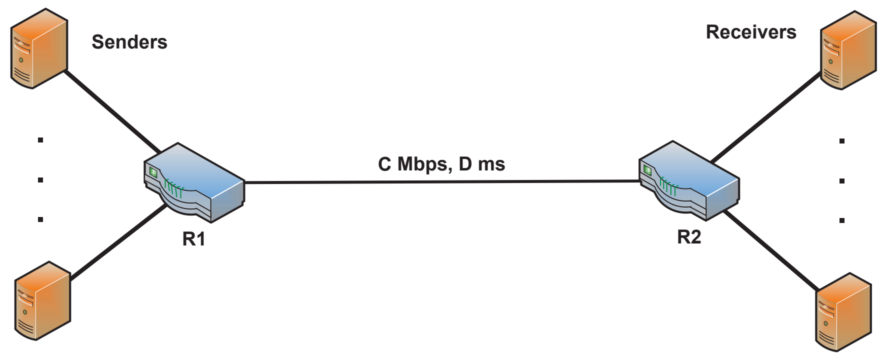
Similarly, in a QCN system, the rates are distributed to maximize:

$$F_B(\vec{R}) = \frac{K\eta}{(K-\eta)^2} \sum_{i \in \mathcal{S}} R_i^T \log{(1 - \frac{K-\eta}{K R_i^T} R_i)} \tag{2.22}$$

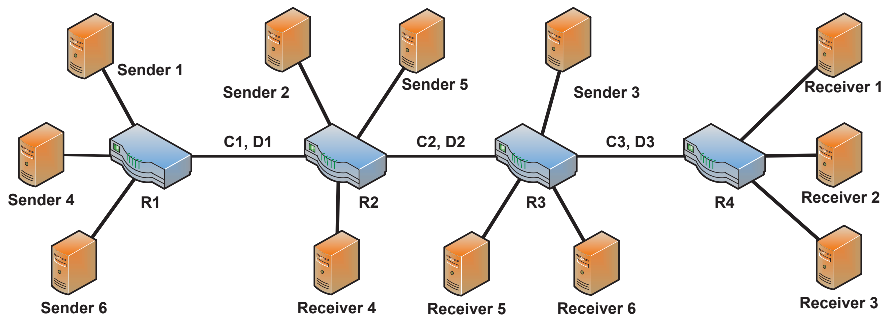
At an equilibrium point, R_i is very close to R_i^T. Hence, it can be observed that in a QCN system, the weight given to R_i is close to $K\eta/(K-\eta)^2 R_i^T \log{(\eta/K)}$ for large R_i, while the weight given to R_i in a FQCN system tends to $-\frac{16 B_L G_d \Psi_{F_b}(i) C_l}{K} \log{(1 + \frac{16 B_L G_d \Psi_{F_b}(i)}{K})}$ as R_i tends to C_l. Thus, FQCN is closer to max-min fairness than to proportional fairness. This conforms to the intuition of FQCN: low rate sources are less likely to receive QCN congestion notification messages, while overrated flows will decrease the sending rate according to the quantized congestion feedback value in the congestion notification messages.

2.6 Performance Evaluation

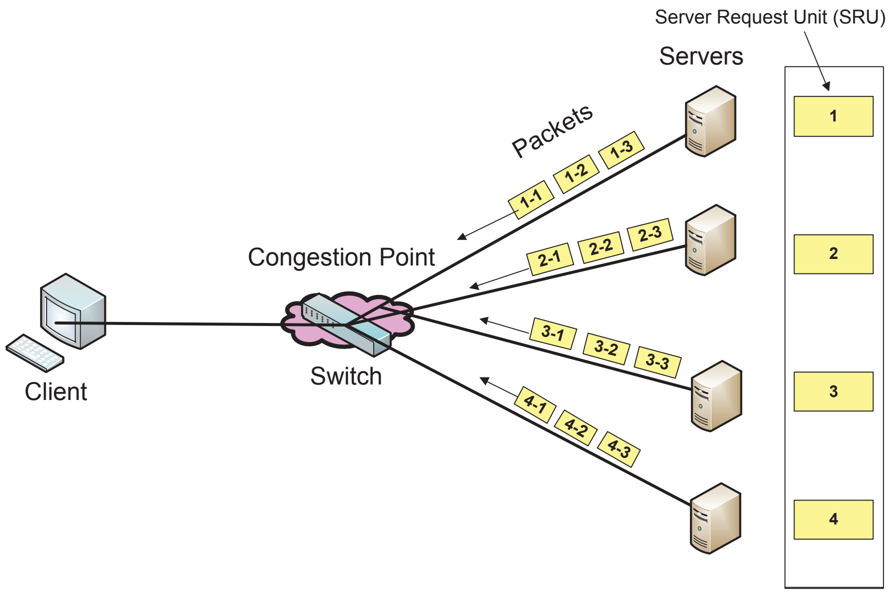
In this section, the performance of FQCN with different traffic dynamics was evaluated by using NS2 [22] under three network topologies, namely, the dumb-bell topology, parking-lot topology and a simple and representative TCP Incast network topology. The performance of FQCN was compared with that of QCN and AF-QCN in terms of fairness and convergence, and the effects of these congestion notification algorithms on TCP Incast. The default QCN configuration is used in our evaluations: w=2, G_d=1/128, T=15 ms, C_T = 5, $B_L = 150$ KB, R_{AI}=5 Mbps and R_{HAI}=50 Mbps when the link capacity of the switch is 10 Gbps, while R_{AI}=0.5 Mbps and R_{HAI}=5 Mbps when the link capacity of the switch is 1 Gbps.



(a) Dumbbell



(b) Parkinglot



(c) A simple and representative TCP Incast network setting with one client requesting data from multiple servers through synchronized reads.

Figure 2.3 Simulation topologies.

2.6.1 Simulation Topologies

The dumb-bell and parking-lot topologies are shown in Figure 2.3(a) and 2.3 (b), respectively. In these two topologies, all the links have 10 Gbps link capacity and 50 μs round-trip time (RTT) delay unless otherwise stated. 150 kilobytes (KB) of switch buffers are used and the equilibrium queue length Q_{eq} is set to 33 KB. Furthermore, the application performance with synchronized reads over TCP was simulated in NS2 to model a typical striped file system data transfer operation to test the effects of QCN, AF-QCN and FQCN on the TCP Incast problem.

TCP Incast [23, 24], also called TCP throughput collapse, has been observed in many data center applications, e.g., in cluster storage [1], when storage servers concurrently respond to requests for a data block, in web search, when many workers respond near simultaneously to search queries, and in batch processing jobs like MapReduce [3], in which intermediate key-value pairs from many Mappers are transferred to appropriate Reducers during the “shuffle” stage. TCP Incast is attributed to having multiple senders overwhelming a switch buffer, thus resulting in TCP timeout due to packet drops at the congestion switch as analyzed in [24–26]. A simple and basic representative network topology in which TCP throughput collapse can occur is shown in Figure 2.3(c). Data is stripped over a number of servers, and stored as a server request unit (SRU) on each server. In order to access one particular data block, a client needs to perform synchronized readings: sending request packets to all of the storage servers containing a fragment of data block for this particular block. The client will not generate data block requests until it has received all the data for the current block. Upon receiving the requests, the servers transmit the data to the receiver through one Ethernet switch almost concurrently. Small Ethernet buffers may be exhausted by these concurrent flood of traffic, thus resulting in packet loss and TCP timeouts. Therefore, TCP Incast may be observed during synchronized readings for data blocks across an increasing number of servers. In this testing scenario, all the links have 1 Gbps link capacity and 100 μs RTT delay and the switch buffer size is 64KB. In order to validate the simulation results, all of these experimental parameters are configured

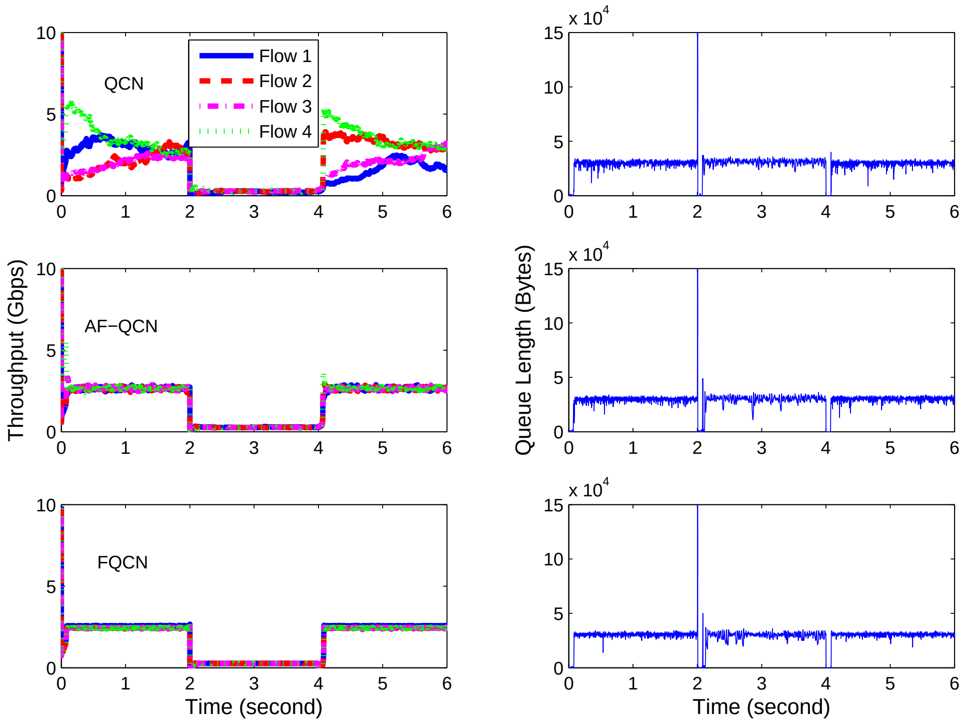


Figure 2.4 The average throughput of four simultaneous static sources (left) and switch queue length (right) in the dumb-bell topology.

as the same as one of those experiments conducted in [24], which explored the TCP Incast problem with various configuration parameters, such as buffer size, SRU size, and TCP variants.

2.6.2 Simulation Results

Backlogged Static Traffic Several experiments were conducted with static backlogged flows in the dumbbell topology and parking-lot topology to validate the fairness improvement and queue length stability of FQCN. The static backlogged traffic sources in our evaluations are simulated with constant bit rate (CBR) traffic flows carried by User Datagram Protocol (UDP), unless otherwise stated.

In the first experiment, four static flows are initiated simultaneously to traverse through the single bottleneck in the dumb-bell topology. The total simulation time is 6

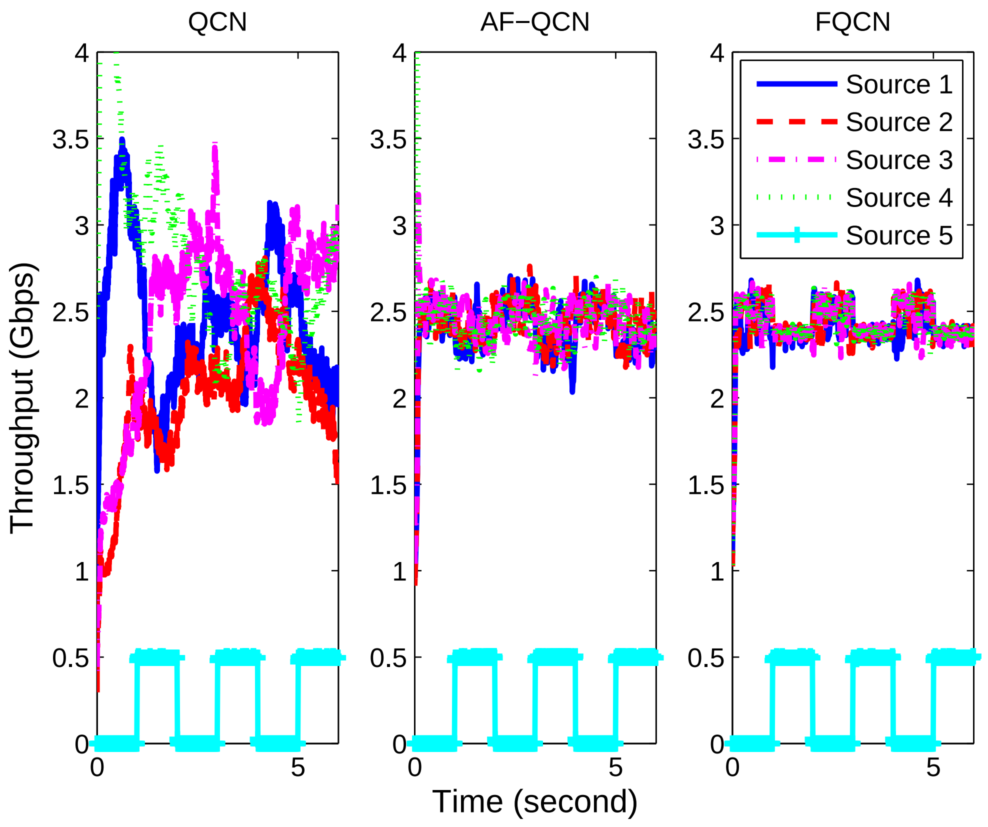


Figure 2.5 The average throughput of four simultaneous static sources and one low rate flow (500 Mbps) in the dumb-bell topology with QCN, AF-QCN and FQCN.

seconds, and the switch service rate is reduced from 10 Gbps to 1 Gbps and changed back to 10 Gbps at 2 and 4 second of the simulation time, respectively. The flow rates of individual flows and the switch queue length for QCN, AF-QCN and FQCN are shown in Figure 2.4. Note that the four sources achieve very different rates by QCN, showing the unfairness of QCN. In this experiment, AF-QCN and FQCN can allocate their equal fair share rates successfully while maintaining the queue length stability. The sending rates with FQCN stabilize to their fair share rates faster and smoother than those of AF-QCN. The standard deviation of sending rates for AF-QCN are 3-4 times larger than that of FQCN.

One more experiment was conducted with four simultaneous static flows and one low rate 500 Mbps CBR source starting periodically as shown in Figure 2.5. The low rate source lasts 1 second each time, and restarts 1 second later. Note that with one low rate

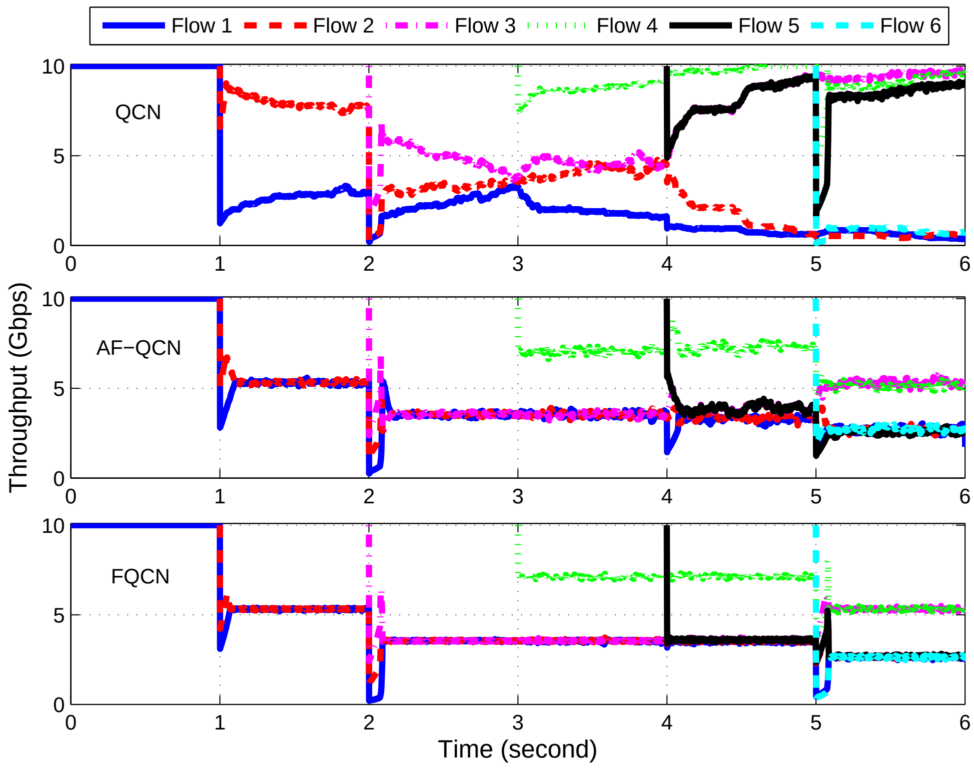
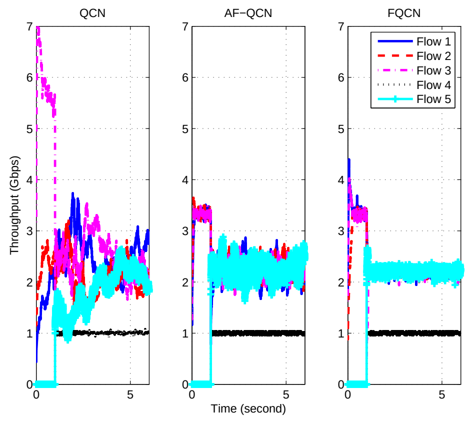


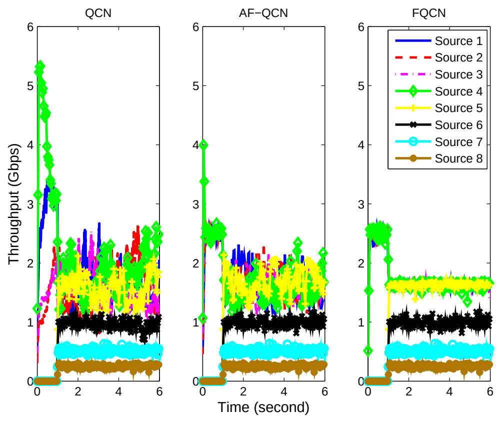
Figure 2.6 The average throughput in the parking-lot topology.

source flow, the total throughput of the bottleneck link can reach its link capacity with all three congestion notification algorithms, QCN, AF-QCN and FQCN. The four high rate sources can share the left over link capacity almost equally fairly with AF-QCN and FQCN. It can further be noticed that FQCN adjusts the transmission rates of four high rate sources and converges to their new rates much faster than AF-QCN when the low rate flow stops or restarts.

In the parking-lot topology, six static source flows are initiated one after another at an interval of 1 second. The sending rates of each individual flow with QCN, AF-QCN and FQCN are shown in Figure 2.6. All of these experiments further confirm the successful allocation of fair share rates with AF-QCN and FQCN while the queue length stability is also maintained which is not shown here. Again, the flow sending rates with FQCN stabilize to their fair share rates faster and smoother than those with AF-QCN. All of these



(a) Mixed static and burst traffic



(b) Mixed static and poisson traffic

Figure 2.7 The average throughputs for individual flows of mix traffic in the dumb-bell topology.

experiments validate our propositions in Section 2.5 that FQCN is stable and the rates converge to maximize the link utilization.

Mixture of Static and Dynamic Traffic Figure 2.7 (a) shows the average throughput of mixed traffic of three static flows and two ON-OFF burst traffic flows in the dumb-bell topology. The average offered traffic loads for these two ON-OFF burst traffic flows are 1 Gbps and 5 Gbps, respectively. All the static flows are initiated at 0 second and the burst traffic flow is initiated at 1.0 second. The "ON" period of the burst traffic flow is determined by completing 10 KB data transfer, and the "OFF" period is chosen to meet the average burst traffic load. The average throughput of the burst flow with lower load is equal to its 1 Gbps burst traffic load since its offered load is lower than its fair share. If the offered burst traffic load is larger than its fair share, its throughput will be limited to its fair share. As seen in the experiment with 5 Gbps burst traffic load, the average throughput for this burst flow is limited to its fair share of 2.25 Gbps. The simulation results in this experiment also validate again that the left over capacity is shared by other three static flows unfairly with QCN and almost equally fairly with AF-QCN and FQCN, and the average throughput is more stable to reach its fair share with FQCN than that with AF-QCN.

Figure 2.7 (b) shows the average throughput of mixed traffic of backlogged static flows and dynamic flows, whose arrival rates are Poisson distributed, in the dumbbell topology with eight source-destination pairs under different congestion notification algorithms. The first four sources are backlogged static flows, and the other four dynamic sources are built up with [8 8 4 2] connections at the flow arrival rates of [250 125 125 125] Mbps and the flow size with Pareto distribution with a mean of 10 KB and a shape parameter of 1.1, respectively. Thus, these four dynamic sources offer [2.0 1.0 0.5 0.25] Gbps traffic load in total, respectively. Simulation results show that the average throughput for each individual dynamic flow is around [1.6 1.0 0.5 0.25], respectively. The average throughput for dynamic source with 2.0 Gbps total traffic load is overloaded and is limited to its fair share of 1.65 Gbps. The other dynamic sources are light-loaded sources whose traffic loads are smaller than their equal fair shares, and therefore the average throughput for these light-loaded dynamic sources are equal to their offered load. Note that FQCN successfully allocates the residual capacity, left over from the light-loaded dynamic sources, equally fair among four backlogged static sources and one overloaded dynamic source, and each individual source quickly converges to and stays stably at its fair share.

Effect of Congestion Notification Algorithms to TCP Incast Figure 2.8 depicts the TCP throughput collapse for a synchronized read application performed on the network shown in Figure 2.3 (c). As a comparison, TCP goodput without any congestion control is also shown in the same figure. In this simulation, a SRU size of 256 KB is chosen to model a production storage system, the default Minimum Timeout Retransmission (RTO) is 200 ms, TCP NewReno implementation with the default packet size of 1000 bytes is used in the simulation, and the equilibrium queue size is set to 14 KB. From the simulation results, it can be observed that without any congestion control, TCP goodput drops very quickly with the increase of the number of servers. Note that QCN and AF-QCN perform poorly in the TCP Incast setup (Figure 2.3 (c)). QCN can delay the occurrence of TCP Incast, but the TCP goodput also decreases with the increase of the number of servers, reducing to around

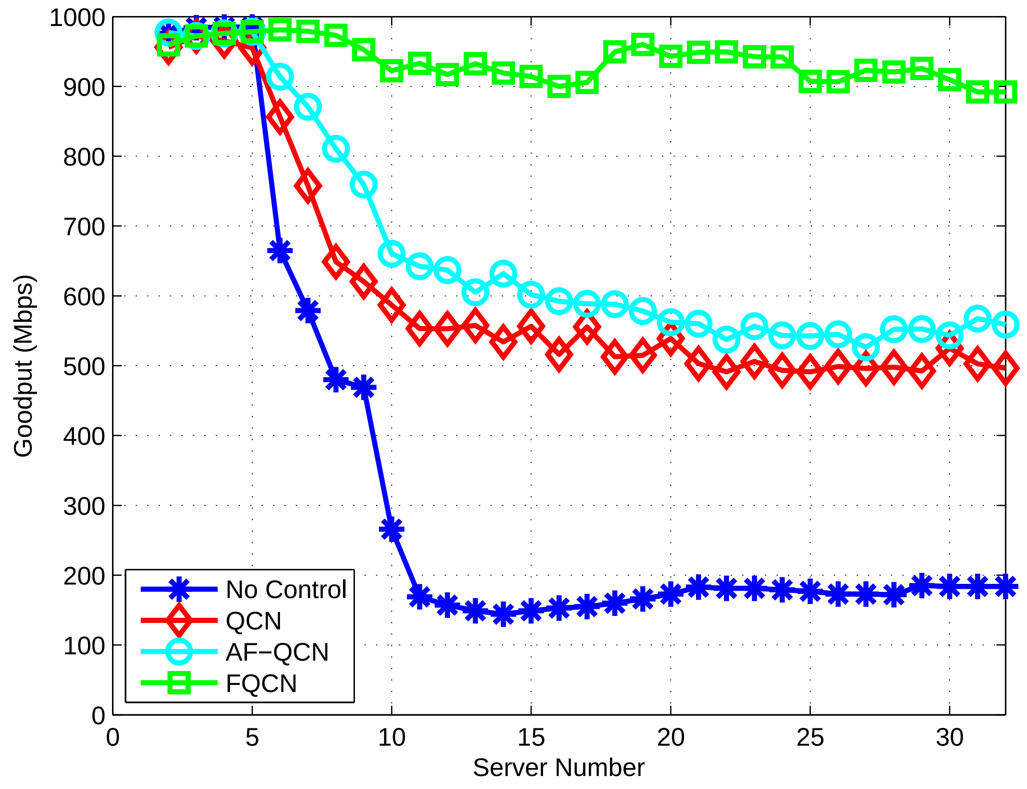


Figure 2.8 TCP throughput for a synchronized reading application.

500 Mbps with 16 servers. AF-QCN achieves a TCP goodput of 50 Mbps more than QCN. On the other hand, FQCN is able to mitigate TCP Incast superbly without much goodput degradation, maintaining a high goodput of around 900 Mbps even with a large number of servers, as shown in Figure 2.8.

The poor performance of TCP throughput with QCN and AF-QCN is attributed to the rate unfairness of different flows. The rate fluctuation of different flows within one synchronous reading request with QCN was examined. Figure 2.9 shows the instantaneous TCP throughput over 1 ms of different flows within one synchronous reading request over eight servers with QCN. The average time to complete one synchronous reading over eight servers is around 25.8 ms, corresponding to about 650 Mbps as shown in Figure 2.8. As shown in Figure 2.9, the time between the first server and the last server having completed their respective data transfers is about 13 ms. Additionally, the sending rates allocated

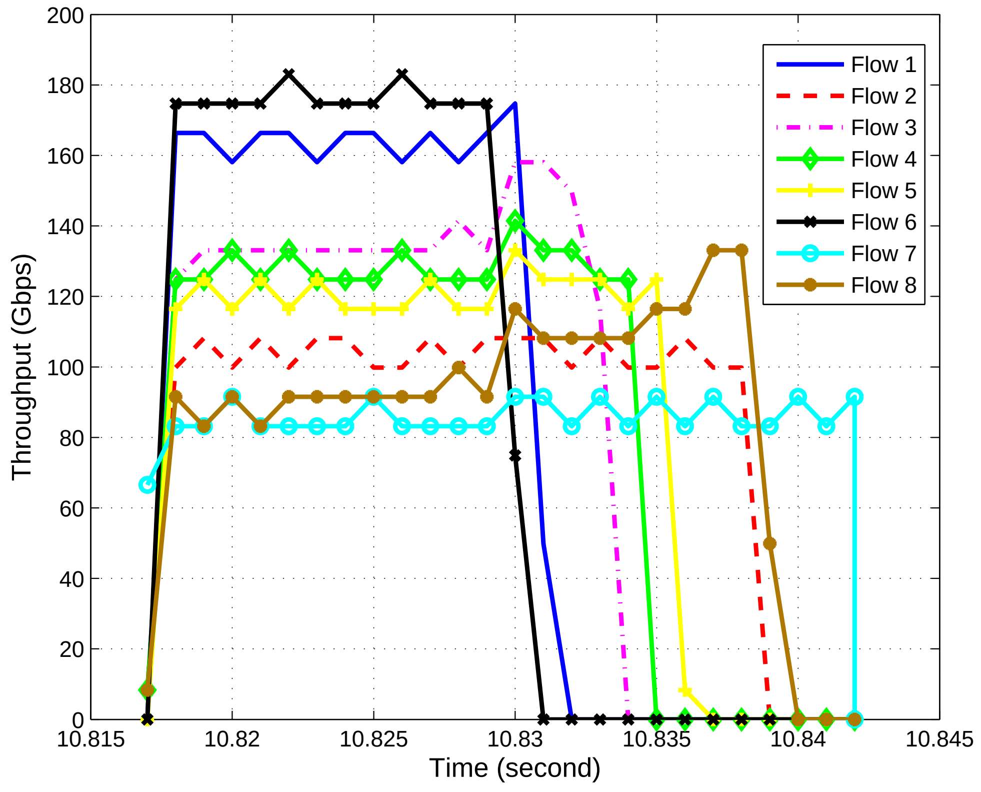


Figure 2.9 The instantaneous throughput of one synchronized reading over eight servers with QCN.

to different traffic flows are quite different from each other. Traffic flows 6 and 7 have higher allocated sending rates than other flows, and more than 30% of the bottleneck link capacity is utilized by these two flows. Thus, they finish the data transfer first, after which other traffic flows attempt to increase their sending rates to recover the available bandwidth previously utilized by traffic flows 6 and 7. It takes several milliseconds to recover this part of available bandwidth by other traffic flows since the departure of traffic flows with relatively higher sending rates; during this period, the bandwidth is not fully utilized. Hence, the unfair rate allocation with QCN results in poor TCP throughput in this TCP Incast setting.

Similar unfair rate allocation to different flows in this TCP Incast setup with AF-QCN can also be observed, but with smaller sending rate variations among different traffic flows and smaller time interval between the first and last server in completing the data transfer than that with QCN in one synchronous reading request. Further, what cause the rate unfairness of different flows in TCP Incast setup with AF-QCN was examined. Similar to QCN, AF-QCN only feeds the congestion notification message back to the source of the sampled packet. It takes some time to let all sources receive the congestion information, and so the later the source gets the congestion notification message, the more likely the source experiences packet drops. As a transmission control protocol, TCP exercises a congestion control mechanism itself. When packet drops occur, TCP slows down its sending rate by adjusting the congestion window size and slow start threshold. AF-QCN only regulates Layer 2 sending rate and attempts to allocate bandwidth fairly among all sources by taking into account of the flow-dependent congestion information in the congestion notification message. However, TCP does not take any flow-dependent congestion information to adjust its congestion window size and slow start threshold. This causes the rate unfairness of different flows in TCP Incast setup with AF-QCN eventually.

The rates of all flows are examined within one barrier synchronous reading request with the FQCN congestion control algorithm, thus facilitating fair sharing of the link capacity among all the source flows because all the overrated flows receive the congestion notification messages at almost the same time.

Different Traffic Weights FQCN can also allocate weighted fair share rates as well as AF-QCN. The flows rates in the dumb-bell topology with FQCN and weight coefficient settings [4, 3, 2, 1] for four static simultaneous flows are shown in Figure 2.10. The larger the weight coefficient, the larger share rate will be allocated to the flow. In this simulation, the maximum sending rate for the first flow whose weight coefficient is set to 4 will be cut down from 10 Gbps to 1 Gbps at the third second of the simulation time. From the simulation results, it can be observed that before the maximum sending rate changes, the flow rates are allocated fairly consistent to their weight coefficients. After the third second, the sending rate of the first flow is limited by its maximum sending rate of 1 Gbps, and

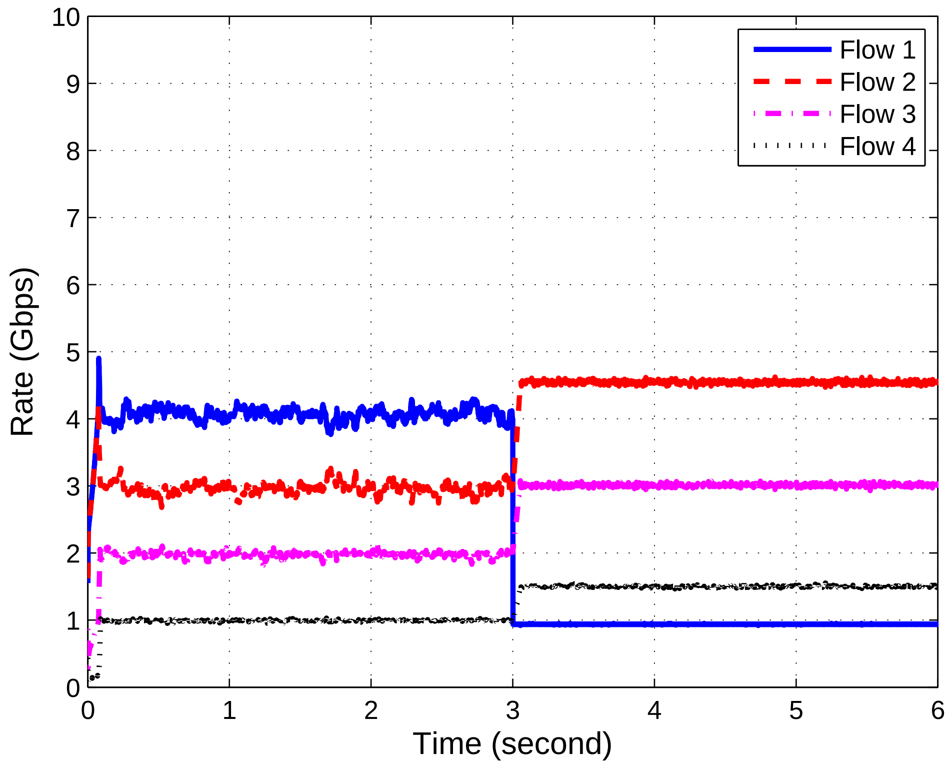


Figure 2.10 Flows rates in the dumb-bell topology with FQCN and weight coefficient settings [4, 3, 2, 1] for four static flows.

the leftover link capacity is shared by other traffic flows fairly consistent to their weight coefficients. Similar results with AF-QCN can also be obtained as those of FQCN in this experiment.

2.7 Summary

In this chapter, an enhanced QCN congestion notification algorithm, called FQCN, has been proposed to improve fairness allocation of bottleneck link capacity among all sharing sources. The performance of FQCN have been evaluated with different traffic dynamics by simulations under three network topologies, namely, the dumb-bell topology, parking-lot topology and a simple and representative TCP Incast network topology. The simulation results have shown that FQCN can successfully allocate the fair share rate to each traffic source while maintaining the queue length stability. As compared to AF-QCN, the flow sending rates with FQCN are more stable with respect to the fair share rates. FQCN can also provide weighted fair share allocation of the bottleneck link capacity. Moreover, FQCN significantly enhances TCP throughput performance with respect to TCP Incast, and achieves better TCP throughput than QCN and AF-QCN in a TCP Incast setting. FQCN performance under other traffic conditions will be further studied in the future. Finally, millions of flows co-existing in a data center network may raise the scalability problem for FQCN due to the need of estimating per-flow information. A distributed flow monitoring system can be helpful to solve this problem, which is left for our future work.

CHAPTER 3

A UNIFIED CONGESTION DETECTION, NOTIFICATION AND CONTROL SYSTEM FOR DATA CENTER NETWORKS

In this chapter, a unified congestion detection, notification and control system for data center networks has been designed to efficiently resolve the network congestions in a uniform solution and to ensure convergence to statistical fairness with "no state" switches simultaneously. The proposed system decouples the congestion control intelligence from switches, and incorporates a distributed potential congestion culprit estimation mechanism and a distributed congestion detection mechanism. In addition, it can easily be incorporated with any congestion notification and control algorithm, and can be deployed for cross-layer congestion control. The fair quantized congestion notification (FQCN) algorithm has been incorporated in the proposed congestion detection, notification and control system, and its performance has been evaluated through extensive simulations, which validate the successful allocation of fair share rates and the maintenance of queue length stability of the proposed congestion control system.

3.1 Introduction

Building an efficient network to interconnect servers within a data center and across multiple data centers is a critically important task for modern data center providers. Congestion detection, notification and control is very crucial to provision such an efficient network.. Currently, most of congestion control mechanisms for data center networks are independently incorporated in different layers, such as Ethernet layer [4–9,15–18], network layer [27–30], or transport layer [31,32].

The Ethernet Layer or Layer 2 congestion notification mechanism [4–9,15–18] is an extremely important technology for data center bridging (DCB) [13] to provide end to end congestion management in switched Ethernet to avoid frame loss. Quantized Congestion Notification (QCN) [7] is the state-of-the-art congestion notification algorithm incorporated into the IEEE 802.1Qau standard for DCB. QCN randomly samples the traffic packets at the switch and feedbacks a global flow-independent congestion status sequentially to a single traffic source deemed as the congestion culprit. This random congestion feedback in QCN prevents the sending rates from converging to statistical fairness [14]. In order to ensure scalability and practical implementation, one of the main requirements and design rationales of IEEE 802.1 for QCN is to enforce the QCN enabled switch to be a "no state" switch, which will not store any per flow state. This requirement also prevents QCN from providing any fairness stronger than proportional fairness, such as max-min fairness. To enhance QCN, approximate fairness QCN (AF-QCN) [17] and fair QCN (FQCN) [15, 16] have been proposed to improve the fairness allocation of link capacity among all shared sources. Both AF-QCN and FQCN require per-flow information, such as flow rate estimate. Millions of flows co-existing in a data center may present the scalability problem for AFQCN and FQCN to estimate per-flow information.

Several traffic engineering solutions in the network layer [27–30] have been proposed to deal with network congestion caused by unbalanced link utilization in data centers. These solutions have been proposed and embodied in software defined networking (SDN) [33], which simplifies the network management by decoupling the control plane from the data plane. The main idea behind these solutions is to balance the traffic load by rerouting the identified elephant traffic flows to resolve network congestion. Even though all the congestion information has already been collected by the SDN controller, traffic engineering solutions are limited to congestion eliminations caused by elephant flows only. Therefore, the detected congestion information may not be fully explored.

TCP is the de facto standard for Internet-based commercial communication networks. However, today's state-of-the-art TCP protocol falls short in data centers and several TCP variants [31,32] have been proposed. Considering the specific traffic characteristics of data centers, such as the interaction between short latency sensitive flows and long background throughput-oriented flows, DCTCP [31] leverages Explicit Congestion Notification (ECN) and extracts multi-bit feedback on congestion in the network from the single bit stream of

ECN marks. In DCTCP, switches mark the ECN bit in the packet if the queue length is over a predefined threshold. Sources estimate the fraction of ECN marked packets, and use this estimate as an indication of the extent of congestion to adjust the congestion window size accordingly. DCTCP can reduce the Ethernet switch buffer occupation, but both TCP sender and receiver are required to be modified. ICTCP [32], a receiver window based congestion control algorithm for TCP at end hosts, was proposed to improve the throughput on incast congestion. ICTCP targets to avoid packet loss caused by buffer overflow, but ICTCP only works for the last-hop congestion.

Such an independent congestion control mechanism may reduce the congestion control efficiency, increase the congestion control overhead, and cannot provide an efficient congestion detection, notification and control over the whole data center network. Primarily, network congestion is detected and notified repeatedly, resulting in the increase of the congestion information messages over the network. In addition, the detected congestion information is not efficiently deployed to resolve network congestion. The congestion control mechanisms deployed on different layers encapsulate the detected congestion information in different congestion notification messages, and therefore, the congestion information detected by congestion control mechanisms cannot be identified and utilized to resolve network congestion uniformly. SDN allows network layer congestion control mechanisms to collect the network congestion information over the whole network. However, such global network congestion information is only utilized for rerouting some specific traffic flows to balance the network traffic and is not fully explored for Ethernet layer and transport layer congestion control mechanisms, which can greatly benefit from such global network congestion information.

Inspired by SDN [33], a unified congestion detection, notification and control system to ensure convergence to statistical fairness with "no state" switches is introduced in this chapter. The proposed system exhibits a number of advantages. First, it decouples the congestion control intelligence from switches, and thus simplifies the implementation and upgrade of the congestion control algorithms. Second, it incorporates a distributed potential

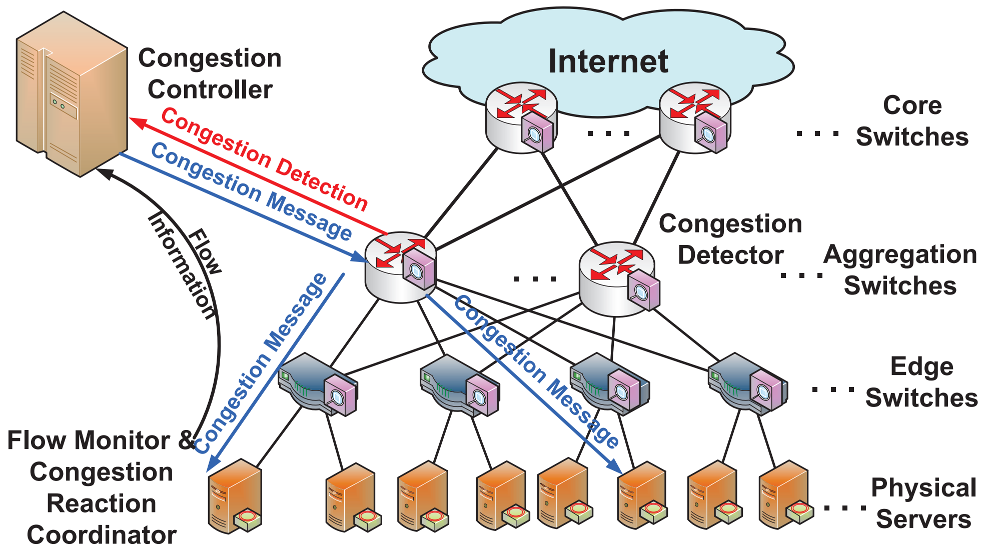


Figure 3.1 The architecture of the proposed congestion detection, notification and control system for data center networks.

congestion culprit detection mechanism. Third, switches are ”no state”, i.e., switches monitor the local congestion status, forward the congestion status to the centralized congestion controller, and do not need to store per-flow information. Also, the proposed congestion control system can easily be incorporated with any congestion notification and control algorithm, regardless whether per-flow information is required or not. Besides, it can be deployed for cross-layer congestion control.

The rest of the chapter is organized as follows. The architecture of the proposed data center congestion detection, notification and control system is described in Section 3.2. The performance of the proposed congestion control algorithm is evaluated in Section 3.3. Section 3.4 concludes the chapter.

3.2 The Architecture of the Proposed Unified Congestion Detection, Notification and Control Framework

The framework architecture of the proposed congestion detection, notification and control system is quite similar to that of SDN. In SDN, a centralized controller calculates the routing paths for traffic flows and helps switches to build routing tables. Figure 3.1 shows the architecture of the proposed congestion detection, notification and control system, which is composed of four components: congestion detectors, a centralized congestion controller, traffic flow monitors, and congestion reaction coordinators. The congestion detectors are implemented at switches to detect the local congestion status through queue monitoring. A traffic flow monitor and a congestion reaction coordinator are deployed at each end host. The traffic flow monitor estimates the sending rates of traffic flows sent out from the attached end host, and informs the flow rates to the centralized congestion controller. The centralized congestion controller collects the congestion information from the congestion detectors and the flow information from the traffic flow monitors. Based on these information, the congestion controller identifies congestion culprits and informs the end hosts of the identified congestion culprits about the congestion status. After the end hosts receive the congestion messages from the centralized congestion controller, the corresponding congestion reaction coordinators will react accordingly to resolve network congestion. As a preliminary example, how to incorporate FQCN [15,16] into the proposed framework is illustrated.

3.2.1 Congestion Detectors

One congestion detector is deployed at each switch to detect the switch congestion. At time t, the congestion detector samples the incoming packets with a sampling probability $p(t)$, and computes the congestion value $F_b(t)$. The sampling probability is initialized to 1%, and is updated after computing the congestion value $F_b(t)$ at each sampling event. Denote $Q_{len}(t)$ and $Q_{len}(t-\tau)$ as the instantaneous queue length in bits of the current sampling event at time t and last sampling event at time $t-\tau$, respectively, where τ is the time interval between two adjacent sampling events. The congestion value $F_b(t)$ consists of a weighted sum of the instantaneous queue offset $Q_{over}(t) = Q_{len}(t) - Q_{eq}$, where Q_{eq} is the equilibrium queue length, and the queue variation over the last sampling interval $Q_\delta(t) = Q_{len}(t) - Q_{len}(t-\tau)$, as defined below:

$$F_b(t) = -(Q_{over}(t) + w * Q_\delta(t)) \tag{3.1}$$

where w is a non-negative constant, taken to be 2 for the baseline implementation.

If $F_b(t)$ is negative, a congestion message containing the value of quantized $|F_b(t)|$, denoted as $\Psi(F_b(t))$, the source of the sampled packet, as well as the identification of the congested link, expressed in the addresses of the two end nodes connecting to the congested link, is sent to the centralized congestion controller; otherwise, no congestion notification message is generated. $\Psi(F_b(t))$ is used to inform the congestion control center how severe the congestion is, the identification of the congested link is used to distinguish where the congestion occurs, and the source of the sampled packet is identified as a congestion culprit if no flow information related to the congested link has been collected by the congestion controller, because the flow information collected at the end servers experiences longer delays than the congestion information collected by switches in general.

At each sampling event, the sampling probability is updated as a function of the calculated congestion feedback value $F_b(t)$ as follows:

$$p(t) = \begin{cases} (1 + \frac{9}{64}\Psi(F_b(t)))\% & (F_b(t) < 0) \\ 1\% & (F_b(t) \geq 0) \end{cases} \tag{3.2}$$

The above equation shows that if there is no congestion ($F_b(t) \geq 0$), the congestion detector checks the congestion status with a probability of 1%; otherwise, the sampling probability is increased as a linear function of the quantized congestion information value $\Psi(F_b(t))$. Since the maximum quantized value of $F_b(t)$ in the default QCN implementation is 64, the maximum sampling probability is 10%.

3.2.2 Traffic Flow Monitors

The traffic flow monitor counts the amount of traffic measured in bytes for each flow originating from the attached end host during a time interval T_s seconds, and estimates the sending rate for each flow. The estimated sending rate for flow i is denoted as B_i, and is updated every T_s milliseconds (ms) as follows:

$$B_i = (1 - \beta)B_i + \beta B_i^{'} \tag{3.3}$$

where $B_i^{'}$ is the newly estimated sending rate of traffic flow i during the last T_s interval, and $\beta \in (0, 1)$ is the weight given to the newly estimated sending rate $B_i^{'}$ against the old estimation of B_i.

If the estimated sending rate of flow i, B_i, is larger than a predefined rate threshold, it is assumed to be a potential congestion culprit. The predefined rate threshold is used to filter out the small size mice traffic flows, i.e., smaller than 1 MB, which account for a large percent of all traffic flows in data center networks [34–36].

The flow monitor periodically estimates the sending rate for each flow originating from the attached end host and updates the flow information of those potential congestion culprits, including the estimated sending rate and the flow identification tuple, which is composed of a couple of values to represent the flow, such as the source and destination addresses, and the source and destination port numbers, to the centralized congestion controller.

3.2.3 The Centralized Congestion Controller

When a congestion notification message is received by the centralized congestion controller, it implements congestion control intelligence to identify the congestion culprits based on the flow information collected by the flow monitors and the congestion information collected by congestion detectors, and informs the congestion status to all identified congestion culprits through multi-casting.

The congestion notification message originated from a switch contains the quantized congestion information value $\Psi(F_b(t))$, the source of the sampled packet, and the identification of the congested link l, which can be used to fetch all the flow information of the potential congestion culprits on this link, denoted as T_l. If the set of T_l is empty, meaning that there is no flow information of the potential congestion culprits related to link l stored at the congestion controller, the intelligent congestion control resumes the QCN algorithm, in which the congestion message will be sent to the source of the sampled packet with $\Psi(F_b(t))$. Otherwise, FQCN [15, 16], a congestion control algorithm based on per-flow information, will be utilized to resolve the network congestion. FQCN identifies the congestion culprits using the flow information of the potential congestion culprits traveling over the congested link l and informs the congestion status to all the identified congestion culprits.

The fair share for each flow i on link l can be estimated as:

$$M_i = \frac{W_i}{\sum_{k \in T_l} W_k} \sum_{k \in T_l} B_k \tag{3.4}$$

where W_i is the weight coefficient for flow i, which can be determined by the traffic class, the source address, the destination address, and etc. A traffic flow i whose estimated sending rate B_i is smaller than its estimated fair share M_i ($B_i < M_i$) is considered as a low rate flow. Otherwise, it will be assigned to the high rate source set $T_l^H = \{i \in T_l | B_i \geq M_i\}$.

The fair share can be fine-grained among the high rate source set T_l^H as:

$$M_i^F = \frac{W_i}{\sum_{k \in T_l^H} W_k} \sum_{k \in T_l^H} B_k \tag{3.5}$$

The congestion culprits can be identified by comparing B_i with M_i^F. A traffic flow i whose estimated sending rate B_i is equal to or larger than its estimated fine-grained fair share M_i^F ($B_i \geq M_i^F$) is considered as an overrated flow, which is deemed as a congestion culprit. All of overrated flows form an overrated flow set $T_l^R = \{i \in T_l | B_i \geq M_i^F\}$.

For each congestion culprit, the quantized congestion feedback value $\Psi_{F_b}(i, t)$ in the congestion message to the source of the congestion culprit i is calculated as follows:

$$\Psi_{F_b}(i, t) = \frac{B_i / W_i}{\sum_{k \in T_l^R} B_k / W_k} \times \Psi(F_b(t)) \tag{3.6}$$

3.2.4 Congestion Reaction Coordinator

The operations of the congestion reaction coordinator are the same as those of the reaction point in QCN [7]. It adjusts the sending rate of the associated traffic source by decreasing the sending rate based on the quantized congestion feedback value contained in the congestion notification message, and increasing the sending rate voluntarily to recover lost bandwidth and probe for extra available bandwidth.

Rate increase Two modules, Byte Counter (BC) and Rate Increase Timer (RIT), are introduced for rate increases. BC and RIT work in two phases, Fast Recover (FR) and Active Increase (AI), based on the state of BC, S_{BC}, and the state of RIT, S_{RIT}, respectively. If the value of S_{BC} is smaller than a predefined threshold value S_T (where S_T is set to 5 in the baseline implementation), BC is in the FR phase; otherwise, BC is in the AI phase. Similarly, RIT staying at the FR or AI phase is determined based on the comparison of S_{RIT} and S_T. Initially, both BC and RIT are in the FR phase and S_{BC} and S_{RIT} are set to 0. S_{BC} is increased by 1 for every B_L or $B_L/2$ bytes transmitted if BC is in the FR or AI phase, respectively. RIT functions similarly as BC. In the FR phase, RIT increases S_{RIT} by 1 for every $T\ ms$ duration. While in the AI phase, S_{RIT} is increased by 1 for every $T/2\ ms$ time duration. Both BC and RIT can raise a rate increase event, when S_{BC} or S_{RIT} is increased, respectively.

When a rate increase event occurs, the current rate $R_C(t)$ and target rate $R_T(t)$ are updated as follows:

$$R_T(t) = \begin{cases} R_T(t) & \text{(both BC \& RIT in FR)} \\ R_T(t) + R_{HAI} & \text{(both BC \& RIT in AI)} \\ R_T(t) + R_{AI} & \text{(otherwise)} \end{cases}$$

$$R_C(t) = \frac{1}{2}(R_C(t) + R_T(t))$$

where R_{AI} is the constant target rate increment if either BC or RIT is in the AI phase, and R_{HAI} is the constant target rate increment if both BC and RIT are in the AI phase.

Rate decrease When a congestion notification message is received by an end host, the congestion reaction coordinator will reduce the sending rate to resolve the network congestion accordingly. The current sending rate $R_C(t)$ is set as the target rate $R_T(t)$ and the current rate is reduced by a factor of $R_C(t)G_d \times \Psi(F_b(t))$ as follows:

$$\begin{aligned} R_T(t) &= R_C(t) \\ R_C(t) &= R_C(t)(1 - G_d \times \Psi(F_b(t))) \end{aligned} \tag{3.7}$$

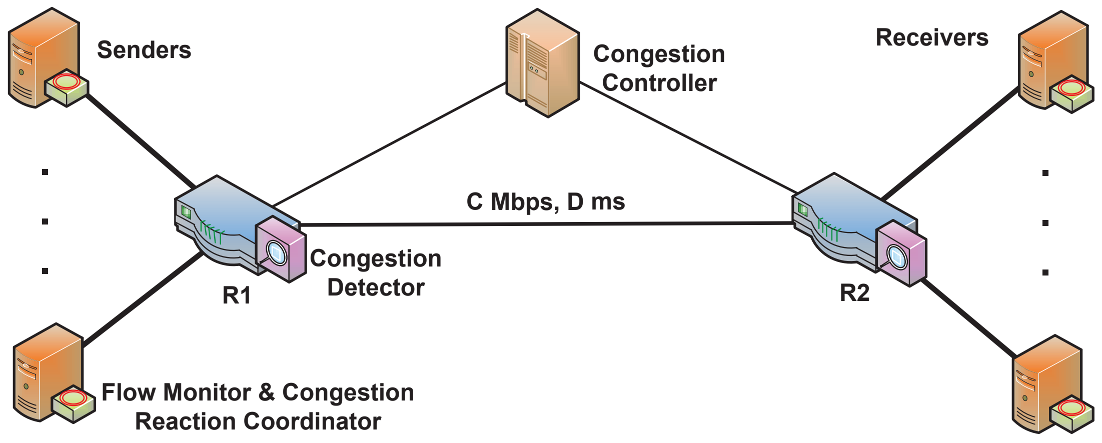
where the constant G_d is chosen to ensure that the sending rate cannot decrease by more than 50%, and thus $G_d * \Psi(F_{bmax}) = \frac{1}{2}$, where F_{bmax} denotes the maximum of $F_b(t)$. In the default implementation, the congestion feedback value $F_b(t)$ is quantized to 6 bits and the maximum quantized value of $F_b(t)$ is 64; thus, G_d is configured to 1/128.

3.3 Performance Evaluation

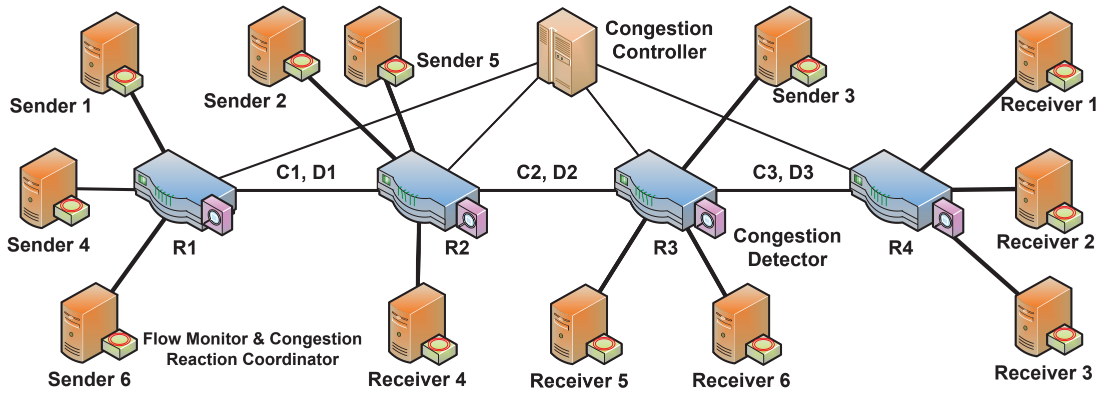
In this section, the performance of the proposed congestion control system which is incorporated with the FQCN algorithm is evaluated by using NS2 under two network topologies, namely, the dumbbell topology and parking-lot topology. The default QCN configuration is used in our evaluations: w = 2, G_d = 1/128, T = 15 ms, S_T = 5, B_L = 150 kilobytes (KB), R_{AI} = 5 Mbps and R_{HAI} = 50 Mbps when the link capacity of the switch is 10 Gbps, while R_{AI} = 0.5 Mbps and R_{HAI} = 5 Mbps when the link capacity of the switch is 1 Gbps.

3.3.1 Simulation Topologies

The dumbbell and parking-lot topologies are shown in Figure 3.2(a) and (b), respectively. In these two topologies, all the links connecting between switches and servers or between switches have 10 Gbps link capacity and 50 μs round-trip time (RTT) delay unless otherwise stated. In each topology, there is a centralized congestion controller, which is connected to each switch to collect the congestion information and implement congestion control. The links between the centralized congestion controller and switches have 1 Gbps link capacity and 100 μs RTT delay. A congestion detector is incorporated at each switch, and a flow monitor and a congestion reaction coordinator are merged to each end server. The flow monitor at each end server updates flow rates every 1 ms. The end servers are traffic sources or traffic sinks. In the dumbbell topology as shown in 3.2(a), all traffic sources, noted as “Senders”, are connected to the switch R_1; while all traffic sinks, noted as “Receivers”, are connected to the switch R_2. In the parkinglot topology, there are six traffic sources and six traffic sinks, denoted as “Sender i” and “Receiver i” ($i \in [1, 6]$),



(a) Dumbbell



(b) Parking-lot

Figure 3.2 Simulation topologies.

respectively, and there is one traffic flow transmitting between a sender and receiver pair (Sender i, Receiver i) ($i \in [1, 6]$).

3.3.2 Simulation Results

Experiments have been conducted with static backlogged flows in the dumbbell topology and parking-lot topology to validate that FQCN incorporated in the proposed system can successfully allocate the fair share rates among the shared sources and maintain the queue length stability. Constant bit rate (CBR) traffic flows are used to simulate the static backlogged traffic in our simulations. 150 KB of switch buffers are used and the equilibrium queue length Q_{eq} is set to 33 KB.

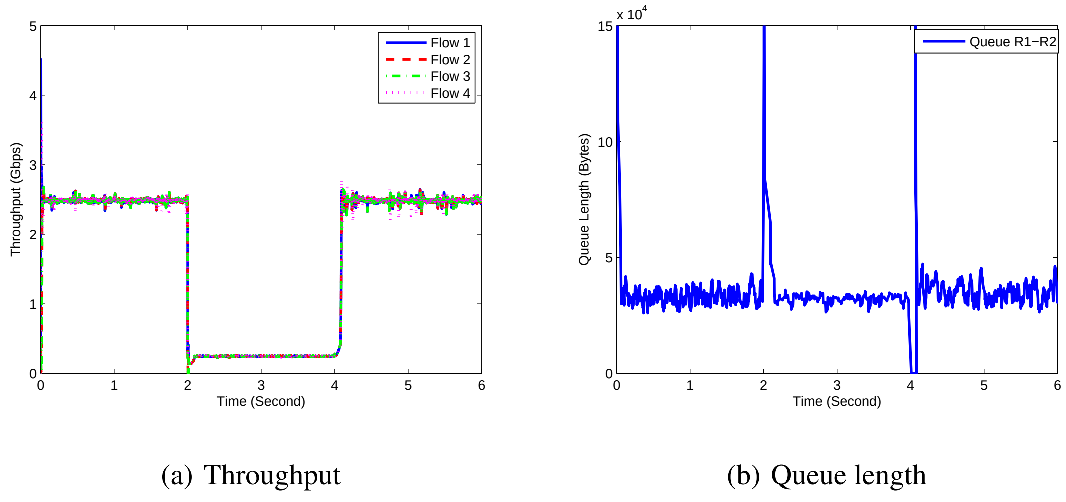
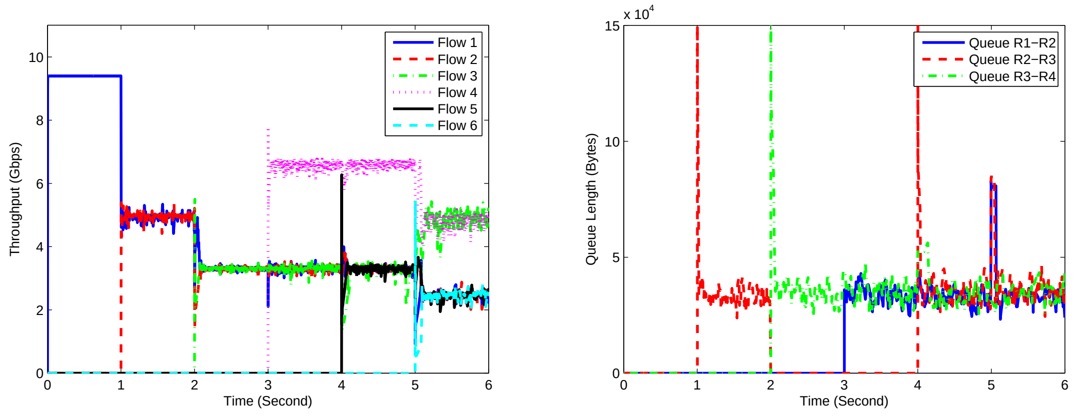


Figure 3.3 The average throughput of four simultaneous static sources (left) and switch queue length (right) in the dumbbell topology.

In the dumbbell topology, four static backlogged traffic flows are initiated simultaneously to traverse through the single bottleneck link between routers R_1 and R_2 as shown in Figure 3.2(a). The switch service rate decreases from 10 Gbps to 1 Gbps and increases back to 10 Gbps at 2 and 4 second of the simulation time, respectively. The flow rates of individual flows and the switch queue length between routers R_1 and R_2 are shown in Figure 3.3(a) and (b), respectively. Figure 3.3(a) shows that at 2 second of the simulation time, the queue between R_1 and R_2 is built up very quickly due to the switch service rate shrunk from 10 Gbps to 1 Gbps. As the queue length increases, the congestion is detected at router R_1 and is informed to the centralized congestion controller, which reacts to the congestion by sending the congestion notification messages to all the identified congestion culprits based on the collected flow information and the congestion information. At 4 seconds of the simulation time, the switch service rate is restored back to 10 Gbps. The congestion reaction coordinator at each end server probes for the available spare bandwidth and increases its source sending rate. Its can also be observed from Figure 3.3(a) that the sending rates of these four backlogged traffic flows are recovered to their original fair share rates about 2.50 Gbps at 4 seconds of the simulation time very quickly. During this period, these four backlogged flows compete with each other for the spare available bandwidth,



(a) Throughput (b) Queue length

Figure 3.4 The average throughput (left) and switch queue lengths (right) in the parking-lot topology.

causing another queue length built up and network congestion between routers R_1 and R_2. However, this congestion is also resolved very quickly and the queue is stabilized at the desired equilibrium point 33 KB again very fast.

In the parking-lot topology, six static backlogged traffic flows, which transmit data from “Sender i” to “Receiver i” ($i \in [1, 6]$), are initiated one after another at an interval of 1 second. The sending rate of each individual flow and the queue length between the routers R_1 and R_2, R_2 and R_3, and R_3 and R_4 are shown in Figure 3.4(a) and (b), respectively. In the first second of the simulation, only flow 1 is activated, which transmits data in the rate of link capacity. No queue is built up at each router. At 1 second of the simulation time, flow 2 is activated, which travels over the switches R_2, R_3 and R_4. Flows 1 and 2 compete the bandwidth of the link between R_2 and R_3 and eventually converge to their fair share rates. During the simulation from 1 second to 2 seconds, the link between R_2 and R_3 is also the bottleneck link and the queue for this link is stabilized to its desired equilibrium point, and no queue is built for the link between R_1 and R_2 and the link between R_3 and R_4. At 2 seconds of the simulation time, flow 3 is activated, which travels over the switches R_3 and R_4. All these three flows 1-3 travel over the link between R_3 and R_4, which is also the bottleneck link during the simulation from 2 seconds to 3 seconds. These three flows converge to their fair share rates of about 3.3 Gbps and the queue between R_3 and R_4 stabilizes to its desired equilibrium point. During the simulation from 3 seconds to 4 seconds, there are two bottleneck links. One is the link between R_1 and R_2, and the other is the link between R_3 and R_4. The queues on these two links are stabilized to their desired equilibrium points during this one second simulation interval. The rates of Flows 1-3 are limited by their fair share rates over the link between R_3 and R_4. Only Flow 1 and Flow 4 travel over the link between R_1 and R_2, while the rate of Flow 1 is limited by its fair share over the link R_3 and R_4, i.e., 3.3 Gbps. Therefore, Flow 4 uses up the leftover link capacity, about 6.7 Gbps. At 4 seconds of the simulation time, another flow, Flow 5, is activated. Flow 5 travels over the switches R_2 and R_3. During the simulation from 4 seconds to 5 seconds, the link between R_1 and R_2, the link between R_2 and R_3, and the link between R_3 and R_4 serve the traffic for Flows 1 and 4, Flows 1, 2, and 5, and Flows 1, 2, and 3, respectively. It is easy to identify that during this one second simulation period, all links between switches are bottleneck links. The newly activated traffic flow, Flow 5, converges to its fair share rate, 3.3 Gbps, very quickly, while the fair share for other four traffic flows are kept the same as last one second of the simulation time. In the last second of the simulation, all traffic flows are activated, and the link between R_1 and R_2, the link between R_2 and R_3, and the link between R_3 and R_4 serve the traffic for Flows 1, 4 and 6, Flows 1, 2, 5 and 6, and Flows 1, 2, and 3, respectively. The rates of Flows 1, 2, 5 and 6 are limited by the their fair share rates over the link between R_2 and R_3, 2.5 Gbps. Flows 3 and 4 use up the leftover link capacity between R_3 and R_4 and between R_1 and R_2, respectively, which is 5 Gbps. Similar to the simulation of the last one second, all links between switches are bottleneck links and the queues on these links converge to their desired equilibrium points, 33 KB.

These experiments validate that FQCN implemented in the proposed system successfully allocates fair share rates to each backlogged traffic flow while maintaining the queue length stabilized round the desired equilibrium queue length. Similar results are obtained in the dumbbell and parking-lot topologies with the RTT delay between the centralized congestion controller and switches increased to 0.1 ms.

3.4 Summary

In this chapter, a unified congestion detection, notification and control system for data center networks has been proposed and developed. The proposed system decouples the congestion control intelligence from switches, employs a distributed potential congestion culprit detection mechanism and a distributed congestion detection mechanism, can be easily incorporated with any congestion notification and control algorithm, and can be deployed for cross-layer congestion control. The FQCN algorithm has been implemented in the proposed system and its performance has been evaluated through simulations. The simulation results have validated the successful allocation of fair share rates and the maintenance of queue length stability with FQCN incorporated in the proposed congestion control system. The performance of other congestion notification algorithms incorporated in the proposed congestion control system will further be studied in the future. Additionally, cross-layer congestion control algorithms for data center networks require further studies within the context of the proposed congestion control system.

CHAPTER 4

HERO: HIERARCHICAL ENERGY OPTIMIZATION FOR DATA CENTER NETWORKS

In this chapter, the power optimization of network elements in data centers is investigated in a hierarchical perspective. Considering the hierarchical architecture of data centers, a two-level power optimization model, namely, Hierarchical EneRgy Optimization (HERO), has been established to reduce the power consumption of data centers by switching off network switches and links while still guaranteeing full connectivity and maximizing link utilization. Given a physical data center network (DCN) topology and a traffic matrix, the proposed two-level power optimizations in HERO, pod-level power optimization and core-level power optimization, are illustrated to fall in the class of capacitated multi-commodity minimum cost flow (CMCF) problem, which is NP-hard. Therefore, several heuristic algorithms based on different switch elimination criteria are designed to solve the proposed HERO optimization problem. The power saving performances of the proposed HERO model are evaluated by several experiments with different traffic patterns. The simulations show that HERO can save power consumptions in data centers effectively with reduced complexity. The optimized power consumptions with HERO are almost the same as those with non-hierarchical model. As compared to the non-hierarchical model, at least 65% and 60% of variables and constraints can be reduced with HERO, respectively. The results also show that among all of the proposed heuristics, switch power consumption heuristic algorithm based on the total power consumption of the core and aggregation switches in the core-level power optimization achieves the minimum power consumption.

4.1 Introduction

In order to provide a reliable and scalable computing infrastructure, the network capacity of data centers is especially provisioned for worst-case or peak-hour workload, and thus data centers consume a huge amount of energy. Report to Congress on Server and Data

Center Energy Efficiency [37] indicated that data centers and servers consumed about 61 billion kilowatt-hours (kWh) in 2006 (1.5% of the total U.S. electricity consumption) for a total electricity cost of about \$4.5 billion, and the electricity usage of data centers has been almost doubled from 2000 to 2006. The data center electricity usage is still growing, but the growth rate has slowed down significantly from 2005 to 2010 due to the economic slowdown and a significant reduction in the actual servers installed as compared to the predicted number of servers to be installed with the improved virtualization techniques. As reported in [38], the total power consumption of data centers in 2010 increases about 56% from 2005 to 2010 for worldwide data centers, and increases only 36% for U.S. data centers instead of doubling, which is significantly lower than that was predicted by the EPA Report to Congress on data centers [37]. The rapid increasing power consumption of data centers exerts great impacts on the environment and society, such as increasing in CO_2 emissions, burden on the power grid, and rise in water usage for cooling [39].

Furthermore, it has been well established in the research literature that the average server utilization is often below 30% of the maximum utilization in data centers [12]. A clear diurnal traffic variation pattern, traffic peaks during the day and falls at night, has been observed in DCNs through production data center trace analysis [36, 40]. The combination of peak load dimensioning and diurnal demand pattern leads to low resource utilization. Unfortunately, today's servers and network elements are not energy-proportional [12, 41–43] since many components incur fixed power overheads when active such as clock, fans, switch chips, and transceivers at low loads. At low levels of workload, servers are highly energy-inefficient because even in the idle state, the power consumed is over 50% of its peak power for an energy-efficient server [12] and often over 80% for a commodity server [41]. Besides, switches are not energy efficient currently either, and they consume 70-80% of their peak power in their idle state [44].

The high operational costs and the mismatch between data center utilization and power consumption have spurred great interest in improving data center energy efficiency. Most efforts to reduce the power consumption of data centers have focused on the power management schemes for server and storage clusters [45–48] and cooling techniques [49–51], which are the largest energy gushers in data centers. With these techniques implemented well in place, network elements, such as routers and switches, become an important, non-negligible and continuously increasing contributor to the overall power consumption in a data center; thus, reducing the power cost of network elements without adversely affecting network performance presents a great challenge.

Several studies [40, 52–54] have investigated the power savings of the network infrastructure in data center networks by switching off some unneeded network devices or by putting them into sleep. In a recent work, a network-wide power manager, ElasticTree [40], was proposed to optimize the energy consumption of DCNs by dynamically optimizing the subset of active network elements, switches and links, to satisfy dynamic traffic loads while still meeting the performance and fault tolerance requirements. ElasticTree optimizes a data center local area network and finds the power-optimal network subset and routing to use by extending the idea of power proportionality into the network domain, as described in [12]. Shang *et al.* [52] solved the energy-saving problem in data center networks from a routing perspective. They established an energy-aware routing model, which minimizes the total number of switches for a given traffic matrix, and proved that the proposed energy-aware routing model is NP-hard, and designed a heuristic algorithm to solve the energy-aware routing problem. Mann *et al.* [53] presented a network power aware framework VMFLow for placement and migration of VMs by taking into account of the network topology as well as network traffic demands to optimize the network power consumption while satisfying as many network traffic demands as possible. However, one assumption in VMFlow does not hold well in practice: at most one VM or a group of VMs, which cannot be separately migrated to different physical host servers, will be placed on one physical server. A more general solution VMPlanner [54] has been proposed to reduce the network elements power consumption by optimizing both virtual machine placement and traffic flow routing. Especially, VMPlanner is a stepwise optimization approach with three approximation algorithms, namely, traffic-aware VM grouping, distance-aware VM-group to server-rack mapping and power-aware inter-VM traffic flow routing. A survey on power consumption in data centers along with solutions has recently been reported in [18].

All of these prior works formulated the power optimization problem for a general network topology, and required a centralized power management controller, which monitors and predicts traffic demands at each network element, and controls the power status of all network elements. As DCNs become larger and larger, the complexity of solving this optimization problem increases rapidly. By investigating the DCN topologies, it can be observed that almost all DCN topologies have hierarchical structures, no matter conventional tree like switching infrastructure [55] or newly proposed data center architectures such as VL2 [56], Portland [57], DCell [58] and BCube [59].

Considering the existing centralized power management mechanisms, a **H**ierarchical **E**ne**R**gy **O**ptimization (HERO) model is proposed for power consumption reduction of data centers in this chapter. Given a data center network topology and a traffic matrix, the possibility of turning off some network elements (i.e., routers, switches and links) hierarchically without violating the network connectivity and QoS constraints is investigated. The main contributions of HERO are described as follows. First, a hierarchical energy optimization model for DCNs is proposed. One of the major advantages of the proposed hierarchical model is that it reduces the algorithm complexity by transforming the whole network power optimization problem into several sub-network power optimization problems. Second, several heuristic algorithms to solve the proposed hierarchical energy optimization model are verified. One more major advantage of the proposed energy optimization model is that different heuristic algorithms at different levels can be adopted.

The rest of the chapter is organized as follows. Section 4.2 presents the background and motivation of the hierarchical approach to reduce the power consumption of data center networks. Details of the proposed hierarchical approach, the network connectivity of the proposed HERO model, and the algorithm complexity are described in Section 4.3. Then, the hierarchical heuristic algorithm is presented in Section 4.4. The performance of the proposed HERO model and heuristic algorithms is evaluated in Section 4.5. Section 4.6 concludes the chapter.

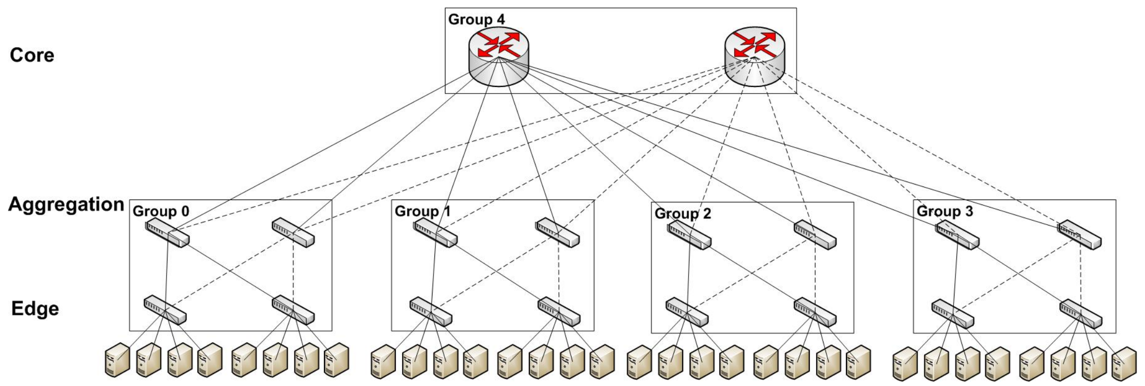
4.2 Background and Motivation

The main idea of the proposed hierarchical power optimization model of the network elements is inspired from the observations and analysis of the data centers' architectures and traffic patterns. In this section, the architectures and the traffic patterns in data centers are discussed in detail.

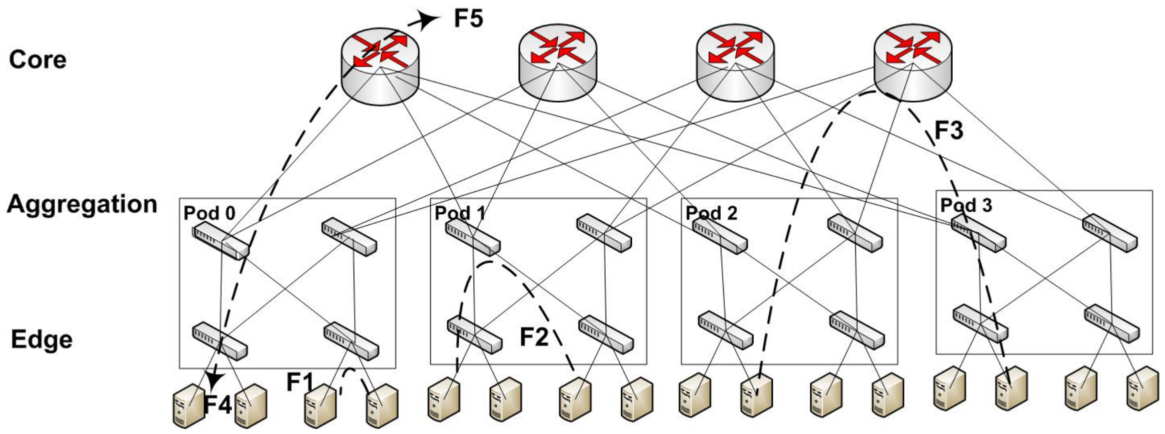
4.2.1 Data Center Topology

Conventional data center networks [55] typically consist of a two- or three-tier hierarchical switching infrastructure. Two-tier architecture only has the core and the edge tiers of switches. In the three-tier architecture, an aggregation switch tier is inserted in the middle of the core and the edge switch tiers. All servers attach to DCNs through edge tier switches. The edge switches and aggregation switches can form several switch groups, and the core switches can form another switch group as shown in Figure 4.1(a). As analyzed in previous works, conventional DCNs incur some well-known problems, e.g., scalability and resource fragmentation, server to server connectivity, and cost. Several new topologies have been proposed for DCNs recently, and they can be divided into two categories, switch-centric topology, e.g., VL2 [56] and Portland [57], and server-centric topology, e.g., DCell [58] and BCube [59].

VL2 [56] is a three-tier architecture and shares almost all of the features with the conventional DCN architecture except the rich connectivity between the aggregation switches and the core switches. PortLand [57] is another three-tier architecture that forms the fat-tree topology. The edge and aggregation switches form a complete bipartite graph, i.e., a Clos graph. A collection of edge and aggregation switches is called a pod in PortLand architecture. Each pod is connected with all core switches, and thus forms a second Clos topology. Thus, each pod can form a switch group, and the aggregation switches and the



(a) Conventional data center network tree-like topology.



(b) Fat-tree data center network topology and traffic pattern.

Figure 4.1 Data center network topologies and traffic categories.

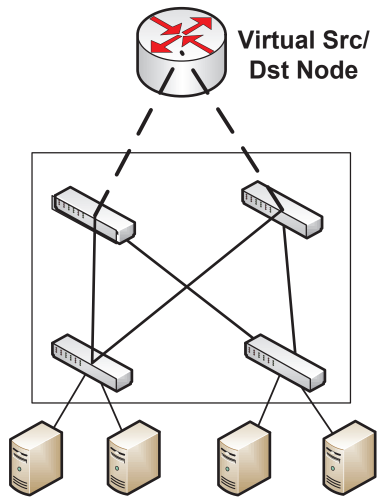
core switches can form another switch group. DCell [58] and BCube [59] are representative modular DCN architectures, which are constructed with new building blocks of shipping containers instead of server racks. Each container houses up to a few thousands of servers on multiple racks within a standard 40- or 20-feet shipping container. The shipping container-based modular data center simplifies supply management by hooking up power, networking, and cooling infrastructure to commence the services, shortens deployment time, increases system and power density, and reduces cooling and manufacturing cost. Both DCell and BCube are multi-level, recursively defined DCN architectures built with mini-switches and severs equipped with multiple network ports. A k-level $DCell_k$ and $BCube_k$ can be constructed recursively with several $(k-1)$-level $DCell_{k-1}$ and $BCube_{k-1}$, respectively. Therefore, each $DCell_{k-1}$ and $BCube_{k-1}$ can form one switch collection and the k-level switches can form another switch group.

As analyzed above, it can be observed that almost all the topologies of data center networks typically consist of a two- or three-tier hierarchical switching infrastructure, including the conventional tree-like switching infrastructure and recently proposed data center architectures. The networking elements, including switches and links, can be organized into several groups.

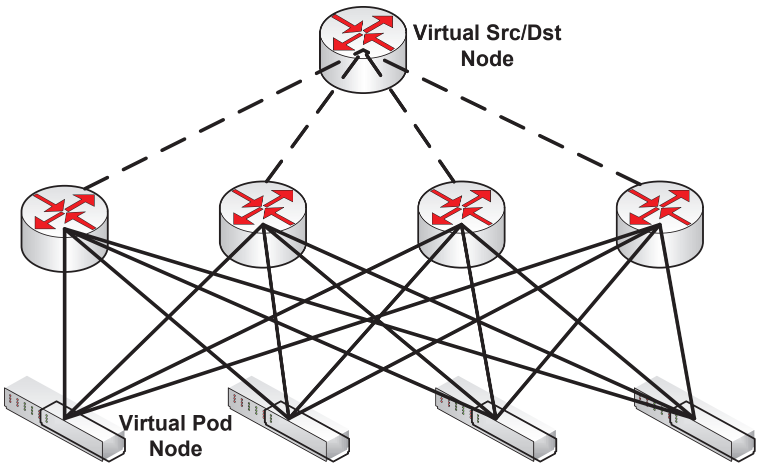
4.2.2 Data Center Traffic Patterns

Traffic in data center networks can be categorized into five classes: intra-edge switch traffic, inter-edge but intra-pod traffic, inter-pod traffic, incoming, and outgoing traffic. For intra-edge switch traffic shown as F1 in Figure 4.1(b), it only requires the connected edge switches and links to be powered on to minimize the power consumption. Inter-edge but intra-pod traffic goes within the same pod, but needs to go through different edge switches as shown as F2 in Figure 4.1(b). Inter-pod traffic is the flows that go through different pods as shown as F3 in Figure 4.1(b). Incoming (F4) and outgoing (F5) are traffic flows that go into and out of data center networks, respectively.

The above observations on hierarchical data center network topologies and data center traffic patterns suggest that a DCN can be divided into several pod-level sub-networks and a core-level sub-network, and the traffic can also be reorganized accordingly. As an example, Figure 4.2 shows the sub-network topologies of a 4-ary fat-tree [60] network as shown in Figure 1(b), which shows a typical 3-tier hierarchical network topology. There are four pods (Pod0-Pod3) in this network topology. Each pod contains edge switches, aggregation switches, and servers connecting to edge switches belonging to this pod. As discussed above, the traffic related to one pod can be reorganized as intra-edge switch traffic, inter-edge but intra-pod traffic, pod-incoming and pod-outgoing traffic. Intra-edge switch traffic and inter-edge but intra-pod traffic are the same as discussed before. The pod-incoming traffic is merged from inter-pod traffic and incoming traffic



(a) Pod-level subgraph



(b) Core-level subgraph

Figure 4.2 Sub-network topologies of a 4-ary fat-tree network.

from outside of the data center network destination to one of the servers belonging to this pod, while the pod-outgoing traffic is merged from inter-pod traffic and outgoing traffic to outside of the data center network originated from one of the servers belonging to this pod. For each pod, a pod-level sub-network can be formed with the original pod, which contains the edge and aggregation switches belonging to this pod, servers connecting to this pod's edge switches and links connecting between this pod's edge switches and aggregation switches or between these edge switches and servers, a virtual node representing the destination of the pod-incoming traffic and the source of the pod-outgoing traffic, and virtual links, each of which connects the virtual node with one of the aggregation switches belonging to this pod as shown in Figure 4.2(a).

Similarly, by abstracting each pod in Figure 4.1(b) to a virtual pod node, a core-level sub-network can be formulated with these virtual pod nodes, the core switches, a virtual source/destination node representing the source of the incoming traffic and the destination of the outgoing traffic, and some virtual links, each of which connects the virtual source/destination node with each of the core switches as shown in Figure 4.2(b). The traffic for this core-level sub-network can be reorganized from the original traffic matrix for the whole data center network by removing the intra-edge switch traffic and inter-edge but intra-pod traffic because the paths for these two types of traffic only travel within one single pod.

The above observations and analysis call for hierarchical energy optimization. The power optimization of data centers can be divided into two levels: core-level and pod-level. The objectives of core-level power optimization are two-fold: to determine the core switches that must stay active to send the outgoing traffic, and the aggregation switches which serve the out-pod traffic in each pod. The objective of pod-level power optimization is to determine the aggregation and edge switches that must be stay alive to flow the intra-pod traffic. The potential benefit of hierarchical energy optimization is to simplify the energy optimization problem of network elements by reducing the number of variables and constraint greatly.

4.3 Hierarchical Energy Optimization Algorithm

As analyzed in the above, the networking infrastructure power optimization of DCNs can be divided into core-level and pod-level network elements power optimization. The optimizer's role in each level is to find the minimum power network subset to meet the performance and fault tolerance goals by powering off the unneeded switches and links. In this section, the hierarchical energy optimization algorithm is described, and the algorithm complexity is analyzed.

4.3.1 Problem Formulation

Given the network topology, the traffic matrix, and the power consumption of each link and switch, the designed core-level and pod-level power optimization can be modeled as two capacitated multi-commodity minimum cost flow (CMCF) [61] problems. Various symbols are summarized in Table 4.1 and the basic assumptions are given as follows:

1) A physical network topology is given. The data center network infrastructure can be modeled by a graph $G = (V, E)$, where V and E are the set of vertices and edges,

respectively. The vertices represent network nodes while the edges represent network links. N and L denote the cardinality of V and E, namely, $N = |V|$ and $L = |E|$, respectively. C_{ij} represents the capacity of link from node i to node j and $\alpha \in [0, 1]$ denotes the maximum link utilization. The core-level and pod-level power optimization problems can be solved based on the core-level sub-graph $G^c = (V^c, E^c)$ and a series of pod-level sub-graphs $G_i^p = (V_i^p, E_i^p)$ $(i \in [1, N_p])$, respectively, where N_p is the total number of pods. The core-level subgraph $G^c = (V^c, E^c)$ as shown in Figure 4.2(b) is formulated in terms of the core switches, the links connecting the aggregation switches and core switches, and the virtual nodes, each of which represents a pod. V^c and E^c denote the set of core-level vertices and edges, and $N^c = |V^c|$ and $L^c = |E^c|$ are the total number of nodes and links in the core-level sub-graph G^c. A pod-level subgraph $G_i^p = (V_i^p, E_i^p)$ $(i \in [1, N_p])$ as shown in Figure 4.2(a) is formulated in terms of a pod and a virtual node representing the destination of the out-pod traffic. V_i^p and E_i^p denote the set of pod-level vertices and edges in the pod-level subgraph G_i^p for the pod p_i $(i \in [1, N_p])$, respectively. And $N_i^p = |V_i^p|$ and $L_i^p = |E_i^p|$ are the total number of nodes and links in the pod-level subgraph G_i^p for the pod p_i.

2) The average amount of traffic exchanged by any source/destination node pair is also given. Denote t_{sd} as the average amount of traffic for the source-target pair $(s, d) \in V \times V$, meaning the traffic that goes from source s to destination d. $T = \{t_{sd}\}$ $(\forall s, d \in V)$ represents the traffic demand matrix which specifies the amount of traffic t_{sd} to be transmitted for each source-destination pair $(s, d) \in V \times V$. The core-level traffic matrix $T^c = \{t_{sd}^c\}$ $(\forall s, d \in V^c)$ and a series of pod-level traffic matrices $T_i^p = \{t_{sd}^{p_i}\}$ $(i \in [1, N_p])$, where p_i $(i \in [1, N_p])$ denotes the ith pod, can be obtained by transforming the traffic demand matrix $T = \{t_{sd}\}$ $(\forall s, d \in V)$. $t_{p_m p_n}^c$ is the average amount of traffic going from source pod p_m $(m \in [1, N_p])$ to destination pod p_n $(n \in [1, N_p])$, where N_p is the total number of pods.

3) The power consumption of each node and link is given. Denote P_i^N and P_{ij}^L as the power consumption of node i, and that of the link between node i and node j, respectively.

Based on the above definitions, the core-level CMCF problem can be formulated as follows: The objective function is to minimize:

$$P^c_{total} = \sum_{i=1}^{N^c}\sum_{j=1}^{N^c} x_{ij}P^L_{ij} + \sum_{i=1}^{N^c} y_i P^N_i \tag{4.1}$$

where P^c_{total} denotes the total power consumption of network switches and links in the core-level sub-graph G^c. $x_{ij} \in \{0,1\}$ and $y_i \in \{0,1\}$ represent the power status of link $(i,j) \in E$ and node $i \in V$, respectively. $x_{ij} \in \{0,1\}$ is a binary variable that is equal to 1 if the link between node i and node j is powered on; otherwise, it is equal to 0. Similarly, $y_i \in \{0,1\}$ is a binary variable that takes the value of 1 if node i is powered on.

Subject to:

1) Flow conservation: commodities are neither created nor destroyed at intermediate nodes.

$$\sum_{i=1}^{N^c} (f^{sd}_{ij})^c - \sum_{i=1}^{N^c} (f^{sd}_{ji})^c = 0, \ \ (\forall s,d,i,j \in V^c, j \neq s,d) \tag{4.2}$$

where $(f^{sd}_{ij})^c$ denotes the amount of traffic flow from node s to node d routing through the arc from node i to node j in the core-level subgraph G^c.

2) Demand satisfaction: each source sends and sink receives an amount of flow equal to its demand.

$$\sum_{i=1}^{N^c} (f^{sd}_{ij})^c - \sum_{i=1}^{N^c} (f^{sd}_{ji})^c = \begin{cases} t^c_{sd} & \forall s,d \in V^c, j = s \\ -t^c_{sd} & \forall s,d \in V^c, j = d \end{cases} \tag{4.3}$$

3) Capacity and utilization constraint: the total flow along each link f^c_{ij} $(\forall i,j \in V^c)$ must be smaller than the link capacity weighed by the link utilization requirement factor α.

$$f^c_{ij} = \sum_{s=1}^{N^c}\sum_{d=1}^{N^c} (f^{sd}_{ij})^c \leq \alpha C_{ij} x_{ij}, \ \ \forall i,j \in V^c \tag{4.4}$$

where f^c_{ij} represents the total amount of traffic flowing on the link $(i,j) \in E^c$ in the core-level subgraph G^c.

4) Switch turn off rule: a node can be turned off only if all incoming and outgoing links are

Table 4.1 Description of Symbols

Parameters	**Description**
$G = (V, E)$	network topology
V	the set of vertices
E	the set of edges
$N = \|V\|$	the cardinality of set V
$L = \|E\|$	the cardinality of set E
$G^c = (V^c, E^c)$	the core-level network topology
V^c	the set of vertices in G^c
E^c	the set of edges in G^c
$N^c = \|V^c\|$	the cardinality of set V^c
$L^c = \|E^c\|$	the cardinality of set E^c
$G_i^p = (V_i^p, E_i^p)$	the pod-level network topology for pod p_i
V_i^p	the set of vertices in G_i^p
E_i^c	the set of edges in G_i^p
$N_i^p = \|V_i^c\|$	the cardinality of set V_i^c
$L_i^p = \|E_i^c\|$	the cardinality of set E_i^c
C_{ij}	the capacity of the link from node i to node j
$\alpha \in [0, 1]$	the maximum link utilization
N_p	the total number of pods
$T = \{t_{sd}\}$	the traffic demand matrix
$T^c = \{t_{sd}^c\}$	the core-level traffic matrix
$T_i^p = \{t_{sd}^{p_i}\}$	the pod-level traffic matrix for subgraph G_i^p
$(f_{ij}^{sd})^c$	the amount of traffic flow from node s to node d routing through the arc $(i, j) \in E^c$
f_{ij}^c	the total amount of traffic flowing on the link $(i, j) \in E^c$
$(f_{ij}^{sd})^{p_m}$	the amount of traffic flow from node s to node d routing through the arc $(i, j) \in E_m^p$
$f_{ij}^{p_m}$	the total amount of traffic flowing on the link $(i, j) \in E_m^p$
P_{total}^c	the total power consumption of G^c
$P_{total}^{p_m}$	the total power consumption of G_m^p
P_{ij}^L	the power consumption of link $(i, j) \in E$
P_i^N	the power consumption of node i
$x_{ij} \in \{0, 1\}$	the power status of link $(i, j) \in E$
$y_i \in \{0, 1\}$	the power status of node $i \in V$

actually turned off.

$$\sum_{j=1}^{N^c} x_{ij} + \sum_{j=1}^{N^c} x_{ji} \leq My_i \ \ , \forall i \in V^c \tag{4.5}$$

where M is twice the total number of links connected to node i in the subgraph G^c.

Similarly, the pod-level CMCF power optimization can also be formulated as follows:

The objective function for pod p_m power optimization is to minimize:

$$P_{total}^{p_m} = \sum_{i=1}^{N_m^p} \sum_{j=1}^{N_m^p} x_{ij} P_{ij}^L + \sum_{i=1}^{N_m^p} y_i P_i^N \tag{4.6}$$

where $P_{total}^{p_m}$ denotes the total power consumption of network switches and links in the pod-level sub-graph G_m^p.

Subject to:

1) Flow conservation:

$$\sum_{i=1}^{N_m^p} (f_{ij}^{sd})^{p_m} - \sum_{i=1}^{N_m^p} (f_{ji}^{sd})^{p_m} = 0, \ \ (\forall s, d, i, j \in V_m^p, j \neq s, d) \tag{4.7}$$

where $(f_{ij}^{sd})^{p_m}$ denotes the amount of flow from node s to node d routing through the arc from node i to node j in the core-level subgraph G_m^p.

2) Demand satisfaction:

$$\sum_{i=1}^{N_m^p} (f_{ij}^{sd})^{p_m} - \sum_{i=1}^{N_m^p} (f_{ji}^{sd})^{p_m} = \begin{cases} t_{sd}^{p_m} & \forall s, d \in V_m^p, j = s \\ -t_{sd}^{p_m} & \forall s, d \in V_m^p, j = d \end{cases} \tag{4.8}$$

where f_{ij}^{sd} denotes the amount of traffic flow from source s to destination d that is routed through the arc between node i and node j in pod p_m.

3) Capacity and utilization constraint:

$$f_{ij}^{p_m} = \sum_{s=1}^{N_m^p} \sum_{d=1}^{N_m^p} (f_{ij}^{sd})^{p_m} \leq \alpha C_{ij} x_{ij}, \ \ \forall i, j \in V_m^p \tag{4.9}$$

where $f_{ij}^{p_m}$ represents the total amount of traffic flowing on the link $(i, j) \in E_m^p$ in the pod-level subgraph G_m^p.

4) Switch turn off rule:

$$\sum_{j=1}^{N_m^p} x_{ij} + \sum_{j=1}^{N_m^p} x_{ji} \leq My_i \ \ , \forall i \in V_m^p \tag{4.10}$$

In order to avoid switching frequently between ON/OFF power states, the switching power penalty of each switch is introduced in our optimization model: the core-level and pod-level objective functions can be extended with Eq. (4.11) and Eq. (4.12), respectively, while the constraints in the core-level and pod-level power optimizations can be kept the same as described above.

$$P'^{c}_{total} \quad = \quad \sum_{i=1}^{N^c} \sum_{j=1}^{N^c} x_{ij} P_{ij}^L + \sum_{i=1}^{N^c} y_i P_i^N + \sum_{i=1}^{N^c} P^s [y_i(1-y_i^-) + y_i^-(1-y_i)] \tag{4.11}$$

$$P'^{p_m}_{total} \quad = \quad \sum_{i=1}^{N_m^p} \sum_{j=1}^{N_m^p} x_{ij} P_{ij}^L + \sum_{i=1}^{N_m^p} y_i P_i^N + \sum_{i=1}^{N_m^p} P^s [y_i(1-y_i^-) + y_i^-(1-y_i)] (m \in [1, N_p]) \tag{4.12}$$

where P^s represents the switching penalty for turning a node off if it was on and for turning a node on if it was off, and y_i^- denotes the power status of node i determined at the last power optimization.

4.3.2 Algorithm Description

A formal description of the HERO algorithm is shown in Algorithm 1. The HERO algorithm can be performed in four steps. In the first step, the power status of the edge switches and edge links connecting the end hosts and edge switches is determined according to traffic matrix T. All edge switches, connecting to any source server or destination server in the traffic matrix T, must be powered on, and others can be powered off. The power status of the core switches and core-level links connecting the core switches and aggregation switches is determined by solving the core-level CMCF optimization problem with the core-level sub-graph G^c, the traffic demand matrix among pods T^c, and the power consumptions of each core switch P_i^N and the core-level links P_{ij}^L. The aggregation switches serve the pod-incoming and pod-outgoing traffics, which

Algorithm 1 Hierarchical Energy Optimization Algorithm

```
Step 1: Determine the power status of edge switches and edge links according to traffic demand matrix $T$.
Step 2: Solve the core-level CMCF optimization problem.
Step 2.1: The power status of core switches and core-level links connecting the aggregation switches and the core switches is decided by solving the core-level CMCF optimization problem.
Step 2.2: The aggregation switches serving the out-pod traffic in each pod are selected with the power status of the core-level links, and the selected aggregation switches are powered on.
Step 3: Solve the pod-level CMCF optimization problem.
for $i = 1$ to $N^p$ do
    Determine the power status of the aggregation switches and the pod-level links connecting the edge switches and the aggregation switches by solving the pod-level optimization problem.
end for
Step 4: In order to provision the whole network connectivity and to meet QoS goals, a merging process is performed.
```

are also the traffic to be considered in the core-level power optimization, and so the aggregation switches with the lowest power consumptions should be selected in the core-level power optimization. However, the aggregation switches are not involved directly in the core-level power optimization because they are contained in the virtual pod nodes. The link connecting an aggregation switch and a core switch is unique, and HERO performs power optimization from the core-level power optimization to the pod-level power optimization, and therefore, the power status of the link connecting one core switch and one aggregation switch in the core-level power optimization also determines the power status of the aggregation switch that this link connects to since if the link is powered up, the switch that it connects to must be powered on anyway. Therefore, in order to guarantee that the aggregation switches with the lowest power consumptions are selected for flowing the pod-incoming and pod-outgoing traffic in the core-level power optimization, the power consumption parameter of the core-level link uses the power consumption of the aggregation switch it uniquely connects to instead of its own power consumption in the core-level power optimization problem. This is based on the observation that the power consumed by one link is much smaller than one switch. The power status of these selected aggregation switches will be used as the input to the pod-level optimization problem in Step 3. The power consumptions of the virtual pod nodes, the virtual source/destination node, and the virtual links connecting the core switches with the virtual source/destination node in the core-level power optimization are assumed to be zero.

Then, in each pod, the power status of the aggregation switches serving intra-pod traffic and that of the pod level links connecting the edge switches and aggregation switches are determined by solving the pod-level optimization problem with the pod-level sub-graph G_i^p, the traffic demand matrix T_i^p in pod p_i, the power consumption of each link P_{ij}^L and node p_i^N in pod p_i, and the power status of the aggregation switches that serve the out-pod traffic determined in Step 2. The aggregation switches selected to be powered on in the second step and the link connecting the selected aggregation switch to the virtual node in each pod are switched on.

Finally, in order to maintain the whole network connectivity, some fault tolerance and QoS guarantees, a complementary process is performed. The basic network connectivity to route traffic defined in the given traffic matrix is ensured by the HERO algorithm, which will be illustrated in the next subsection. From the HERO algorithm, it can be observed that the power status of switches and links optimized with HERO is determined by the given traffic matrix. In most of the cases, the HERO optimization model maintains the whole network connectivity, but in some special cases, the whole network connectivity could be broken. For example, all the traffic flows in a traffic matrix can be classified into intra-edge traffic or inter-edge but intra-pod traffic, meaning that no traffic will travel through any one of the core switches. Therefore, given such a traffic matrix, the optimal solution obtained by the optimization problem in HERO will put all core switches into the idle state, and thus all core switches can be turned off and the connection between the data center and the Internet outside the data center is broken.

Similarly, the connection between a pod and other pods in the data center or the connection between a pod and the Internet outside the data center may be broken if the power status of switches and links in a pod is determined by solving the HERO optimization problem with the given traffic matrix only. In addition, more switches and links might be activated to meet some QoS and fault tolerance goals. The balance between reducing power consumption by turning off part of network elements and turning down QoS and fault tolerance is another research topic and is beyond the scope of this chapter. In this chapter, two basic rules are implemented in the complementary process. One is that at least one core switch is powered on. If none of the core switches is turned on, one core switch is randomly selected to be powered on. Another rule is that at least one aggregation switch that can connect to one active core switch must be turned on in each pod. If no such aggregation switch is turned on in one pod, the one which connects to a powered-on core switch will be switched on, and the link connecting this aggregation switch and the core switch can also be powered on or kept off since switching on a link is much faster than switching on a router or a switch.

4.3.3 Connectivity Certification

The connectivity between the servers and edge switches is maintained since the power statuses of edge switches and edge links are determined according to the traffic matrix directly. The connectivity between the edge and aggregation switches in each pod and the connectivity between the aggregation and core switches are guaranteed by the pod-level optimization and the core-level optimization, respectively. The optimal solutions of the pod-level optimization and the core-level optimization may activate different aggregation switches. Thus, the connectivity for out-pod traffic might be broken. In order to ensure

Table 4.2 Comparison of Algorithm Complexity

Parameters	**Non-Hierarchical**	**HERO-core**	**HERO-pod**
N_{nodes}	$5K^2/4$	$K^2/4+K+1$	$K+1$
N_{links}	$3K^3/4$	$K^3/4+K^2/4$	$K^2/2+K/2$
N_{var}	$3K^3/2 * N_D + 3K^3/4+5K^2/4$	$(K^3+K^2)*N_D^c/2+K^3/4+K^2/2+K+1$	$(K^2+K)*N_D^{p_i}+K^2/2+3K/2+1$
N_{con}	$3K^3/2+5K^2/4*(N_D+1)$	$(K^3+K^2)/2+(K^2/4+K+1)*(N_D^c+1)$	$K^2+K+(K+1)*(N_D^{p_i}+1)$

K denotes the degree of the fat-tree structure, and N_D, N_D^c and $N_D^{p_i}$ denote the total number of traffic demands for the entire network, the total number of core-level traffic demands, and the total number of pod-level traffic demands in pod p_i, respectively.

the connectivity for out-pod traffic, the links connecting the aggregation switches activated in the core-level optimization and the edge switches in each pod together with these edge switches are required to be activated in the pod-level optimization. To realize this, the power status of the aggregation switches determined in the core-level optimization is input to the pod-level optimization. If the leftover capacity of the active aggregation switches is not enough to accommodate the intra-pod inter-edge traffic, more aggregation switches will be activated in the pod-level optimization. Therefore, the connectivity of out-pod traffic flows is ensured by introducing the power status of aggregation switches obtained in the core-level optimization as the initial condition for the pod-level optimization.

4.3.4 Algorithm Complexity

The algorithm complexity of the CMCF optimization problems increases with the increase of the total number of variables and the total number of constraints. As described in Section 4.3.1, the total number of variables in the CMCF power optimization problem can be expressed as the sum of the product of the total number of traffic demands and twice the total number of links considering two directions of communication, the total number of nodes, and the total number of links as follows:

$$N_{var} = N_{links} \times 2 \times N_{demands} + N_{links} + N_{nodes} \tag{4.13}$$

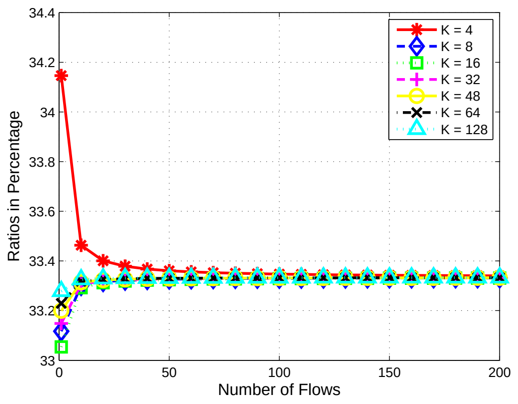
where N_{var}, N_{nodes}, N_{links}, and $N_{demands}$ denote the total number of optimization variables, nodes, links, and traffic demands, respectively.

The total number of constraints in the CMCF power optimization problem can be expressed as the sum of twice the total number of links, the product of the total number of nodes and the total number of demands, and the total number of nodes, by the following equation:

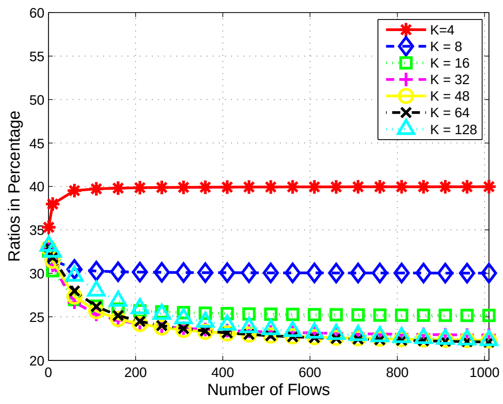
$$N_{con} = N_{links} \times 2 + N_{nodes} \times (N_{demands} + 1) \tag{4.14}$$

where N_{con} represents the total number of constraints.

Table 4.2 compares some major parameters related to the algorithm complexity of the CMCF optimization with a K-ary fat-tree topology. As shown in Figure 4.1(b), a K-ary fat-tree DCN is built up with $5K^2/4$ K-port switches and the edges switches and aggregation switches are constructed to form K pods, each of which consists of $K/2$ edge switches and $K/2$ aggregation switches. The edge and aggregation switches form a complete bipartite graph in each pod. Therefore, the total numbers of nodes and links in pod-level optimization are $K + 1$ and $K^2/2 + K/2$, respectively. There are $(K/2)^2$ K-port core switches, and each core switch has one port connected to each of k pods, while each pod is connected to all core switches, and thus, if each pod is virtualized as a node, the core switches and these virtual nodes form a second bipartite graph. So, the total numbers of nodes and links in the core-level optimization are $K^2/4 + K + 1$ and $K^3/4 + K^2/4$, respectively. The total numbers of switches and links of the whole K-ary fat-tree network are $5K^2/4$ and $3K^3/4$, respectively. The algorithm complexities of hierarchical and non-hierarchical energy optimization problems can be compared in



(a) The ratio of the total number of variables.



(b) The ratio of the total number of constraints.

Figure 4.3 The ratios of the total number of (a) variables and (b) constraints between hierarchical and non-hierarchical energy optimization algorithms for the fat-tree network topology.

terms of the ratio of the total number of variables and the ratio of the total number of constraints. Since the total numbers of links and nodes in each pod are much smaller than those in the core-level, and the core-level and pod-level optimization problems can be solved serially, the algorithm complexity of the core-level power optimization problem dominates the algorithm complexity in the hierarchical power optimization model. Hence, the ratio of the total number of variables and the ratio of the total number of constraints between the hierarchical and non-hierarchical energy optimization algorithm in a K-ary fat-tree DCN can be expressed as:

$$\rho_{var} = \frac{K^3/2\times N_D^c + K^3/4 + (K^2/4+K)}{3/2K^3 * N_D + 3/4K^3 + 5/4K^2} \tag{4.15}$$

$$\rho_{con} = \frac{K^3/2 + (K^2/4+K)*(N_D^c+1)}{3/2K^3 + 5/4K^2*(N_D+1)} \tag{4.16}$$

where ρ_{var} and ρ_{con} denote the ratio of the total number of variables and the ratio of the total number of constraints, respectively. N_D and N_D^c denote the total number of the traffic demands of the entire network and the core-level traffic demands, respectively.

Figure 4.3 presents the ratios of the total number of variables and the total number of constraints between hierarchical and non-hierarchical energy optimization algorithms with different values of K and the total number of flows. Under different K and N_D values, it is obvious that the ratios of the total number of variables and the ratios of the total number of constraints between the hierarchical and the non-hierarchical model are smaller than 35% and 40%, respectively, implying that at least 65% and 60% of variables and constraints in the CMCF optimization problem can be reduced with HERO as compared to the non-hierarchical power optimization model, respectively. The ratio of the total number of constraints decreases with the increase of parameter K with the same number of flows. The results shown in Figure 4.3 are the worst case because the ratio values are calculated under the condition that the total number of the core-level flows is the same as that of the entire network, while in fact the total number of the core-level flows is usually much smaller than that of the whole network.

In general, there are multiple independent pods in a data center. Since the pod-level optimization problems can be solved in parallel, the total computational time to obtain the optimal solution of the HERO model is roughly the sum of the computational time to solve the core-level optimization problem and one pod-level optimization problem. As compared to the non-hierarchical optimization model, the computational time to obtain the optimal solution of the HERO model can also be cut down because both the total number of variables and the total number of constraints are reduced in the optimization problems of the HERO model.

4.4 Heuristic Algorithms

The proposed HERO energy optimization for data centers falls in the class of the CMCF problem. CMCF is NP-hard, and so exact methods can only be used to solve trivial cases. Simple greedy heuristics of node optimization and link optimization were proposed [62] by trying to switch off an additional network node or link. An improved version of this heuristic algorithm was reported in [63] by explicitly considering the power consumption amount of the devices. The basic idea is to iterate through the node set or link set by sorting the nodes or links according to their power consumption in the node optimization or in the link optimization stage. An energy-aware greedy heuristic routing algorithm for data center networks was presented in [52]. The basic idea of this heuristic routing is to gradually eliminate the lightest-loaded switch from those involved in the routing, based on the total throughput of flows carried by each active switch. The problem of this heuristic is that it does not take the switch power consumption model into consideration while different switch models may co-exist in a data center.

In this chapter, several simple greedy heuristic algorithms based on different switch elimination criteria are designed to solve the hierarchical optimization problem for large data center networks. The proposed energy-efficient heuristic algorithm is based on the observation that the power consumed by one link is much smaller than one switch. The switch elimination criterion could be switch throughput, switch power consumption, and switch power efficiency, that gradually eliminates the lightest loaded switch based on the total throughput of flows carried by each active switch, the switch with the highest maximum power consumption per port, and the lowest power-efficient switches, respectively. The power efficiency of a switch is defined as the total throughput of flows carried by the switch divided by its power consumption. The basic idea of the proposed heuristic algorithms is to try to turn off the core switches and core-level links iteratively as well as the aggregation switches and pod-level links iteratively in the core-level and pod-level power optimization, respectively, so that as few switches and links as possible are powered on in DCNs to meet the traffic demands and QoS goals.

All switches and links are assumed to be powered on initially, and the traffic is routed as normal, and then the switches and links are selectively powered off hierarchically in the core-level and pod-level energy optimizations. In the core-level heuristic, the core switches are sorted based on different criteria, and then the selected core switches are attempted to be switched off by checking if the traffic can be flowed through other active core switches; if so, they are turned off. Based on the throughput-based criterion and switch power efficiency criterion, the core switches are sorted in the ascending order, while the core switches are sorted in the descending order if the switch power consumption criterion is adopted. Therefore, the core switch with the lowest throughput, the lowest power efficiency, or the highest power consumption model is eliminated first. A similar heuristic is applied to the pod-level optimization except that it tries to turn off aggregation switches in each pod. As an improvement to the switch power consumption based criterion, another switch power consumption based criterion is proposed by sorting the core switches in the descending order of the total power consumption of each core switch and its connecting aggregation switches in the core-level power optimization.

4.5 Performance Evaluation

In this section, the optimal power consumptions of network elements of the HERO model and those of the non-hierarchical optimization model are compared first. Then, the power saving performance of different heuristic algorithms is investigated. The relationship of the power consumptions and the network utilization is further studied with a diurnal traffic variation over a day. Also, the effects of the power penalty when powering on or off network elements in the power optimization model are examined.

Three different switch power consumption models investigated in [40] are referenced to build a DCN in each evaluation experiment. Each switch selects its switch model randomly (uniform distribution) from these three types. In [40], the power consumptions of three 48-port different switch models with all ports staying at the idle state are measured as 151 W, 133 W and 76 W, respectively. Also, a 1/3 extra power consumption required to turn on all ports is observed in [40]. A positive linear relationship between the switch power consumption with all ports staying at the idle state and the number of switch ports is assumed in our evaluations, meaning that a switch consumes more power with the increase of the number of switch ports.

4.5.1 Traffic Patterns

Currently, the commercial data center traffic traces are not available in the public domain, but tremendous variations in the data center communication matrix over space and time have been observed and reported in [34–36, 56]. According to their reports, data center traffic exhibits a clear diurnal variation pattern, in which traffic peaks during the day and falls at night; most flows (90%) in DCNs are transactional and small in size ($<$1MB), such as search results; the majority of traffic (90%) are transferred in a small fraction of elephant flows, such as backups and back-end operations; a significant fraction of traffic in data centers (80% for cloud data centers and 40-90% for university and private enterprise data centers) are transferred within a rack. Intra-rack traffic amount is larger than inter-rack traffic greatly. The majority of traffic flows last under a few hundreds of milliseconds. In the absence of commercial DCN traffic traces, several traffic patterns were generated to verify the power saving performance of HERO.

1. Random Elephant: an end host sends large elephant flows to any other end host equally likely in DCN.
2. All-to-All Data Shuffle: every end host transfers a large amount of data to every other end hosts.
3. Staggered (P_{out}, P_{edge}, P_{pod}): a source host sends traffic out of a DCN with probability P_{out}, to an end host which is connected by the same edge switch as the source host with probability P_{edge}, and to an end host which is within the same pod as the source host but connected by different edge switches with probability P_{pod}, and to the rest of the end hosts with probability $1 - P_{edge} - P_{pod} - P_{out}$.
4. Diurnal Traffic Variation: the data center traffic variation over one day exhibits a clear diurnal pattern, in which traffic peaks during the day and falls at night.

4.5.2 Simulation Results

HERO vs. Non-hierarchical Model The traffic patterns “Random Elephant” and “All-to-All Data Shuffle” are two extreme cases with large elephant traffic flows only and with small traffic flows only, respectively. In order to compare the optimal power savings of the

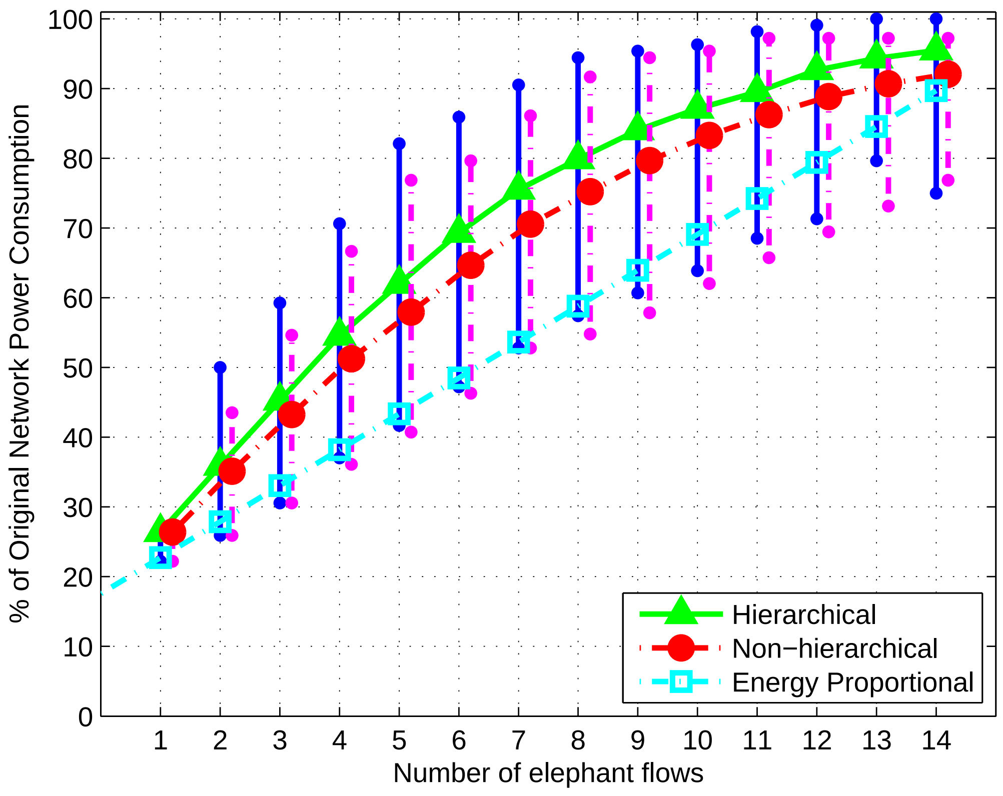


Figure 4.4 The power consumptions of a 4-ary fat-tree DCN with different number of elephant traffic flows.

proposed HERO model and those of the non-hierarchical optimization model, CPLEX [64] is used to obtain the solutions of the CMCF problems in the evaluations with these two traffic patterns.

Since current commodity switches are not power proportional to work loads (even more than 80% to its peak power consumption at idle state) and the elephant flows dominate the traffic volume in DCNs, the performance of the HERO model with elephant flows only was investigated first, the traffic demand of which for each source-destination pair will consume the whole edge link capacity. Figure 4.4 shows the average power consumption of a 4-ary fat-tree network with different numbers of traffic flows. For each number of traffic flows, 1000 simulations are performed with randomly generated traffic flows, and the power consumption is normalized with respect to the maximum power consumption of

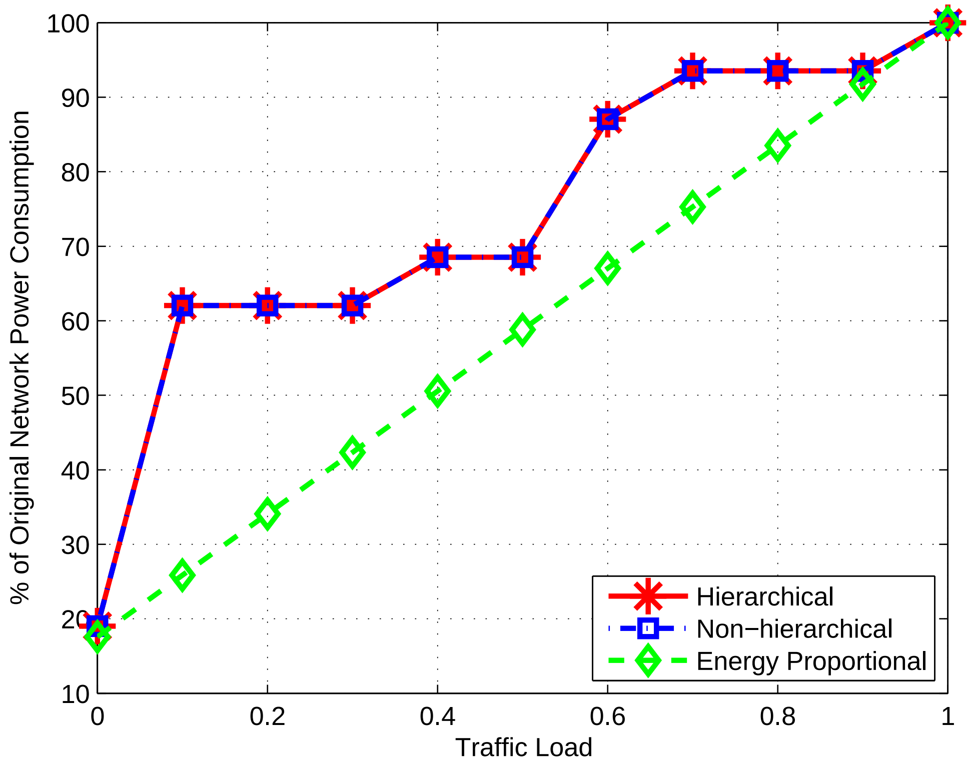


Figure 4.5 The power consumptions of a 4-ary fat-tree DCN with all-to-all data shuffle under different traffic loads.

the whole network. The power consumption of the non-hierarchical power saving model and traffic-load proportional power consumption model are also plotted for comparison. The non-hierarchical power saving model takes the whole DCN topology as an input to the optimization problem and does not consider the hierarchical property of the DCN topology. ElasticTree [40] is a typical example of the non-hierarchical power optimization model. The traffic-load proportional power consumption model is a primary design goal for power optimization schemes, and thus this model provides the lower bound of power consumptions under different traffic loads. Such an energy-proportional model ideally consumes no power when idle, nearly no power when very little traffic is transmitted through the network and gradually more power as the traffic increases.

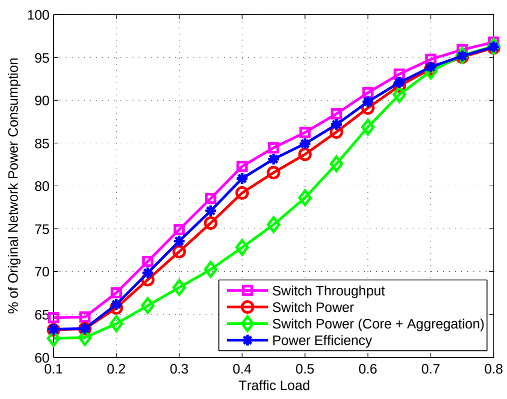
As shown in Figure 4.4, smaller than 5% more of the maximum power consumption of the whole network is consumed in HERO than that of the non-hierarchical model.

Owing to the QoS and fault tolerance protection rules implemented in the complementary procedure, more about 20% of the maximum power consumption of the whole network is needed as compared to the traffic-load power proportional model under the worst case. The lower bound and upper bound power consumption under different numbers of elephant flows are also shown in Figure 4.4, which reflects the influence of source and destination locations on the power consumption. Since no aggregation and core switches are required to be powered on for intra-edge switch traffic flows, the power consumption under some specific number of traffic flows is minimized if all the flows are intra-edge switch traffic. Similarly, the power consumption is maximized if all the flows are inter-pod traffic.

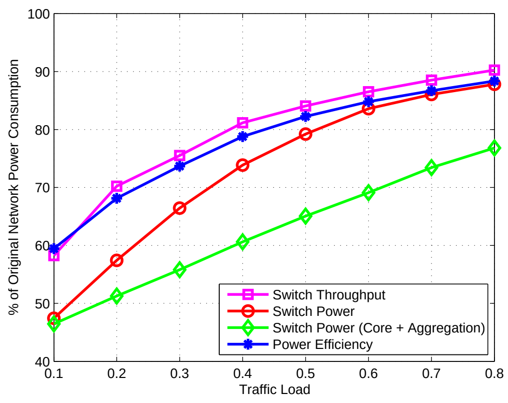
A data shuffle is an expensive but necessary operation for some data center operations. Figure 4.5 shows the normalized power consumption of a 4-ary fat-tree network with all-to-all traffic under different traffic loads. With all-to-all traffic, any host will have a traffic flow to any other end host. The traffic load generated at any host will be distributed to other hosts uniformly. As shown in Figure 4.5, the power consumptions of HERO and the non-hierarchical model are almost the same under different traffic loads. The power consumption at low traffic load is much higher than that of the traffic-load power proportional model because every edge switch is active with all-to-all traffic.

Besides, in these two evaluations, it can also be observed that the computational time to obtain the optimal solution of the HERO model is about half of the computational time to solve the non-hierarchal power optimization problem.

Performance of Heuristic Algorithms The staggered traffic pattern is a mix of intra-edge traffic, inter-edge but intra-pod traffic, inter-pod traffic, and outgoing traffic as discussed in Section II.B, and it is generated to emulate typical data center traffic patterns based on the published work [34–36,56]. With the staggered traffic pattern, experiments are conducted to study the DCN power consumptions with different heuristic algorithms. The power consumptions of four heuristic algorithms under different traffic load with staggered traffic pattern $(0.2, 0.4, 0.24)$ in a 4-ary and 16-ary fat-tree DCN are shown in Figure 4.6(a)



(a) 4-ary fat-tree DCN



(b) 16-ary fat-tree DCN

Figure 4.6 The power consumptions of different heuristic algorithms in a fat-tree DCN with staggered $(0.2, 0.4, 0.24)$ traffic.

and Figure 4.6(b), respectively. Traffic demands are generated randomly. Around 20% of the traffic leaves the data center, and 50%, 30% and 20% of the remaining traffic are intra-edge switch traffic, inter-edge but intra-pod traffic, and inter-pod traffic within the data center, respectively.

As shown in both figures, throughput-based and power-efficiency based switch elimination criteria consume almost the same power. Switch power consumption method based on the total power consumption of the core and aggregation switches in the core-level power optimization, labeled with “Switch Power (Core + Aggregation)”, achieves the minimum power consumption among these four heuristics. At low utilization, about 37% and 53% of the original DCN power consumption can be saved by using this heuristic algorithm in a 4-ary and 16-ary fat-tree DCN, respectively, which are quite close to the corresponding 38% and 60% optimal power savings analyzed in [40]. Also, a linear relationship between the power consumption and the traffic load in the range of 20% - 70% of the maximum traffic load can be observed using this heuristic algorithm. Therefore, the power consumption achieved by this heuristic algorithm is very close to that of the optimal solutions of HERO and non-hierarchical optimization model. Another phenomenon of this heuristic algorithm in a 16-ary fat-tree DCN is also observed: the

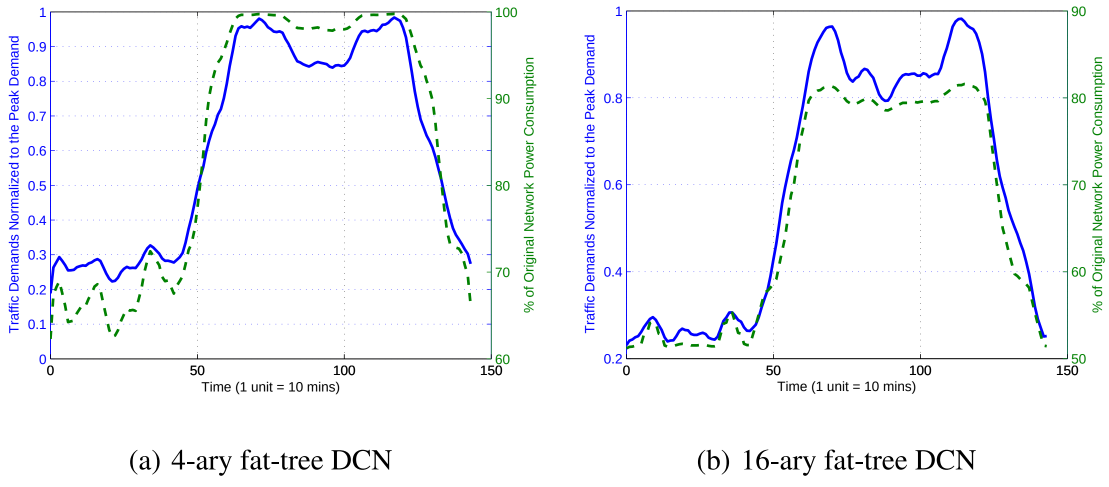


Figure 4.7 The power consumptions over a day with a diurnal traffic variation pattern in a fat-tree data center network with staggered $(0.2, 0.4, 0.24)$ traffic.

power consumptions of network elements optimized with HERO under different network utilizations in this evaluation with the staggered traffic pattern (0.2, 0.4, 0.24) are between the power consumptions of network elements optimized by ElasticTree [40] with 50% far traffic (which needs to transmit through one of the core switches) and 50% middle traffic (which needs to transmit through one of the aggregation switches, but does not need to transmit through a core switch) and those obtained by ElasticTree with all middle traffic. This is due to the staggered traffic pattern $(0.2, 0.4, 0.24)$ used in this evaluation.

Diurnal Traffic Variation and Switching Power Penalty The performance of HERO with a diurnal traffic variation over a day is further studied. The power consumptions with the proposed HERO model over one day in 10-minutes intervals with a diurnal traffic variation pattern with staggered traffic pattern $(0.2, 0.4, 0.24)$ in a 4-ary and 16-ary fat-tree DCN are depicted in Figure 4.7(a) and Figure 4.7(b), respectively. In this evaluation, the heuristic algorithm based on the total power consumption of the core and aggregation switches in the core-level power optimization is used. From these two figures, it can be seen that the power consumptions of data center networks also show a clear diurnal pattern and follow the network utilization curve.

In all of the previous power optimization evaluations, the power penalty when powering on or off network elements are ignored. Such total power penalty was investigated in a 4-ary fat-tree data center network over a one-day diurnal traffic variation with the HERO model and its extended optimization model, which considers the switching power penalty of each switch, respectively. In this evaluation, several experiments were conducted with different settings to the switching power penalty parameter P^S in Eq. (4.11) and (4.12) in the range of $[10\%, 50\%]$ of the switch's power consumption. The results show that in a 4-ary fat-tree data center network, the total switching power penalty in a day with HERO is more than five times larger than that of its extended optimization model which takes the switching power penalty into account. It can also be observed that by taking the switching power cost into the power optimization model, the unnecessary switching power penalty can be minimized.

4.6 Summary

With the increase of scale of DCNs, existing centralized power management schemes to reduce the power consumptions of network elements in DCNs do not scale well. Inspired from the hierarchical architectures and traffic patterns of data centers, the HERO model has been proposed to reduce the power consumption of network elements without violating the connectivity and QoS constraints. In fact, the power optimization of networking elements by switching off the idle switches or links can be divided into core-level and pod-level network elements power optimization. Given a physical DCN topology, the traffic matrix, and the power consumption of each link and switch, the designed core-level and pod-level power optimization problems in the proposed HERO model fall in the class of CMCF problem, which is NP-hard. Several heuristic algorithms based on different switch elimination criteria have been designed to solve the hierarchical optimization problem for large data centers. The switch elimination criteria could be switch throughput, switch power consumption, and switch power efficiency. The basic idea of the proposed heuristic algorithms is to try to turn off the core switches and core-level links iteratively as well as the aggregation switches and pod-level links iteratively in the core-level and pod-level power optimization, respectively.

The performance of the proposed HERO model has been evaluated by generating several traffic patterns, including random elephant, all-to-all data shuffle, staggered (P_{out}, P_{edge}, P_{pod}) and diurnal traffic variation. The simulation results show that the proposed HERO model can achieve optimized power consumptions of a DCN similar to that of the non-hierarchical model but with much less computational efforts. The algorithm complexities of HERO and non-hierarchical energy optimization problems have been compared in terms of the total number of variables and the total number of constraints in a fat-tree DCN. As compared to the non-hierarchical energy optimization model, at least 65% and 60% of variables and constraints can be reduced with the HERO model, respectively. The simulation results show that among all of the proposed heuristic algorithms, switch power consumption heuristic algorithm based on the total power consumption of the core and aggregation switches in the core-level power optimization achieves the minimum power consumption since it considers the power consumption of core switches and aggregation switches in a whole instead of separately.

CHAPTER 5

HETEROGENEITY AWARE DOMINANT RESOURCE ASSISTANT HEURISTICS FOR VIRTUAL MACHINE CONSOLIDATION

In this chapter, several heterogeneity aware dominant resource assistant heuristic algorithms for virtual machine (VM) consolidation are described. The proposed heuristic algorithms explore the heterogeneity of the VMs' requirements for different resources and the heterogeneity of the PM's resource capacities, and utilize the dominant resource (i.e., the resource corresponding to the largest element in the vector of the VM's resource demand or the physical machine's resource capacity) to assist VM consolidation. The performance evaluations using the synthetic workloads validate the effects of the proposed heterogeneity aware heuristics on VM consolidation. The proposed heuristics can achieve quite similar consolidation performance as dimension-aware heuristics with almost the same computational cost as those of the single dimensional heuristics.

5.1 Introduction

Virtual machine consolidation is an important technique for efficient usage of server resources to reduce the total number of servers, thus potentially resulting in reducing the power consumption of data centers. VM consolidation is often modeled as vector bin packing problems, which are well-known to be NP-hard [65]. Therefore, heuristic algorithms are usually used to solve this type of problems [66–68] in practice.

First fit decreasing (FFD) and best fit decreasing (BFD) are the most popular heuristic algorithms to solve the VM consolidation problem. Verma *et al.* [66] investigated a power and migration cost aware application placement framework, referred to as pMapper, to minimize power and migration costs while meeting the performance guarantees in heterogeneous server clusters. They also proposed an improved FFD heuristic algorithm, namely, min power parity (mPP), to map VMs to physical machines (PMs). The basic idea of mPP is to allocate VMs to the most energy-efficient PM, thus resulting in the least power increase per unit increase in resource utilization. Beloglazov and Buyya [67] applied BFD for VM placement in their proposed energy efficient resource management system for virtualized cloud data centers. Similar to mPP, a modified best fit decreasing (MBFD) heuristic algorithm [68] was designed to consolidate VMs onto the most energy-efficient PMs. The main difference between mPP and MBFD is that mPP sorts PMs in the descending order of power efficiency, while MBFD does not.

Besides FFD and BFD, a minimum bin slack based heuristic, namely, incremental power aware consolidation (IPAC), was proposed in [69]. For a given server, IPAC allocates a subset of unallocated VMs to maximize the server's CPU utilization constrained by the server's resource capacity. As shown in [69], IPAC provides a better solution in terms of power consumption than FFD does, but the computational complexity to solve the minimum slack problem may limit its application in practice. Computational intelligence [70, 71], such as genetic algorithms [72–74] and simulated annealing [75], has also been deployed to improve the VM consolidation performance. The VM consolidation performance can be improved with genetic algorithms or simulated annealing over FFD. However, these algorithms are computational expensive, and generally take more time to reach the optimal result.

In this chapter, several heuristic algorithms are described. The proposed heuristic algorithms explore the heterogeneity of the VMs' requirements for different resources and the heterogeneity of the PM's resource capacities, and utilize the dominant resource (i.e., the resource corresponding to the largest element in the vector of the VM's resource demand or the PM's resource capacity) to assist VM consolidation. The dominant resource demand of a VM and the dominant resource capacity of a PM captures the primary request of a VM and the primary resource of a PM, respectively. These two characteristics have not been considered in the previous heuristic algorithms for VM consolidation.

The remainder of this chapter is organized as follows. In the next section, the VM placement problem is introduced. Some popular heuristics for VM placement are described in Section 5.3. Several proposed dominant resource aware heuristics are presented in

Section 5.4. The performance of the proposed heuristic algorithms for VM placement is evaluated in Section 5.5. The summary of the chapter is presented in Section 5.6.

5.2 Virtual Machine Consolidation Problem

Most works published in the research literature formulate the VM consolidation as an optimization problem to minimize the number of active servers with constraints imposed by server capacities, service level agreements, etc. Assume that K VMs $\{V_1, \cdots, V_K\}$ are requested by clients and assigned to a data center consisting of M distinct host physical machines $\{M_i\}$ $(i \in [1, M])$. Each PM M_i can be characterized by a d-dimensional resource capacity vector $C_i = [C_{i1}, \cdots, C_{id}]$ $(i \in [1, M])$. Each dimension corresponds to a different resource such as CPU, memory, network bandwidth, or disk bandwidth. Similarly, each VM is characterized by a d-dimensional resource request vector $R_k = [r_{k1}, \cdots, r_{kd}]$, where r_{kj} $(j \in [1, d])$ is the VM V_k's resource demand for dimension j. The VM consolidation algorithm determines how to allocate these K VMs on M PMs, denoted by $x_{ij} = \{0, 1\}$ $(i \in [1, K], j \in [1, M])$ which is the binary variable indicating whether the VM V_i is assigned to PM M_j. If the VM V_i is assigned to PM M_j, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

Based on the above definitions, the goal of the VM consolidation is to place the VMs on as few as possible active PMs, while at the same time ensuring that the total resource demands of VMs allocated on any active PM does not exceed its resource capacity across any dimension. If PM M_j has been assigned one or more VMs, this PM is said to be active, $y_j = 1$; otherwise, it stays in the idle state, $y_j = 0$. Thus, the objective function of the VM consolidation algorithm is to minimize the total number of active PMs as:

$$\min \textstyle\sum_{j=1}^{M} y_j \tag{5.1}$$

subject to

1) Resource capacity constraints: for each resource of one PM M_j, the total quantity utilized by VMs allocated to this PM cannot exceed its corresponding resource capacity

weighed by its maximum resource utilization factor γ_{jk} ($k \in [1, d]$).

$$\textstyle\sum_{i=1}^{K} r_{ik} \cdot x_{ij} \leq \gamma_{jk} \cdot C_{jk} \quad (\forall j \in [1, M], \forall k \in [1, d]) \tag{5.2}$$

2) Placement guarantee constraints: each VM can only be assigned to one PM.

$$\textstyle\sum_{j=1}^{M} x_{ij} = 1 \quad (\forall i \in [1, K]) \tag{5.3}$$

3) Correlation between the binary status of PMs y_j with VMs allocation x_{ij}: for each PM M_j, its binary status y_j should not be smaller than the binary allocation status of VM V_i allocated to PM M_j x_{ij}, since any one of the VM V_i assigned to PM M_j will result in $y_j = 1$.

$$y_j \geq x_{ij} \quad (\forall i \in [1, K], \forall j \in [1, M]) \tag{5.4}$$

5.3 Heuristics for Virtual Machine Consolidation

In this section, some popular heuristic algorithms for virtual machine consolidation will be reviewed and discussed. Based on the scale used for VM assignment, heuristic algorithms for VM consolidation can be classified into two categories: single dimensional heuristics and dimension aware heuristics.

5.3.1 Single Dimensional Heuristics

The basic idea behind the single dimensional heuristics is mapping the vector of PM's resource capacities and the VM's resource demands into a single scalar, which will be used to perform VM consolidation while ignoring the relationships across dimensions.

First Fit Decreasing (FFD) FFD heuristic sorts the VMs and PMs in the non-increasing order of their "size". It could be the normalized resource demand or capacity in any one of the dimensions (e.g., CPU, memory), or the weighted sum of normalized resource demand or normalized resource capacity across all dimensions, or the product of normalized demand or normalized capacity across all dimensions [76]. The FFD heuristic algorithms based on these different sorting rules are denoted as *FFDSDim*, *FFDSum*, and *FFDProd*, respectively, throughout this chapter. Hence, the VM "size" can be defined as follows:

$$Size(V_i) = \begin{cases} r_i & (\textit{FFDSDim}) \\ \sum_{j=1}^{d} w_j r_{ij} & (\textit{FFDSum}) \\ \prod_{j=1}^{d} r_{ij} & (\textit{FFDProd}) \end{cases} \tag{5.5}$$

where $\{w_j\}$ ($j \in [1, d]$) denotes the weight coefficients for different resources. Similarly, the PM's "size" can be defined by replacing the vector of the VM's resource requests with the vector of the PM's resource capacity.

Starting from the largest VM in size, FFD iteratively assigns VMs to PMs. For each VM, FFD also starts from the largest PM in size to iteratively check the PM's residual capacity, which is defined as the PM's resource capacity subtracting the total resource demands of VMs assigned to this PM. The VM will be packed to the first PM which has enough residual capacity, implying that the residual capacity of this PM is larger than the VM's resource demand across any dimension. If none of active PMs can host the VM, the largest idle PM will be activated.

Best Fit Decreasing (BFD) Similar to FFD, the BFD heuristic also sorts the VMs in no-increasing "size", and iteratively assigns each VM to active PMs starting from the largest VM. The main difference between BFD and FFD is that BFD allocates the VM to PMs based on some rules, such as Least Full First (LFF) and Most Full First (MFF). With LFF, the VM will be assigned to the active PM with the least resource utilization. On the other hand, with MFF, the VM will be assigned to the active PM with the highest resource utilization.

5.3.2 Dimension Aware Heuristics

The main drawback of single dimensional heuristics is that the vectors of the PM's resource capacities and the VM's resource requests are mapped into a single scalar, and thus the PMs' complimentary requirements for different resources are not considered during the VM assignment procedure. Contrary to the single dimensional heuristics, the dimension aware heuristics attempt to take advantages of the PMs' complimentary resource requirements across all resource dimensions.

DotProduct Singh *et al.* [77] proposed a load balancing scheme, called VectorDot, for handling the multi-dimensional resource constraints in data centers. Basically, VectorDot utilizes the dot product $\sum_i^d w_i r_i h_i(t)$ between the vector of a PM's residual capacities at time t, $H(t) = \{h_1(t), \cdots, h_d(t)\}$, and the vector of the VM's resource requirement elements $R_i = \{r_{i1}, \cdots, r_{id}\}$ across all dimensions to choose the target PM for VM placement, where w_i is the weight parameters calculated in the same manner as described in the FFDSum heuristic. This idea can also be used for VM consolidation, which is denoted as the DotProduct heuristic algorithm. Given a PM, DotProduct selects and allocates the VM that maximizes the dot product $\sum_i^d w_i r_i h_i(t)$ while not violating the PM's capacity constraints. The main idea of DotProduct is to select and assign a VM on a given PM for which the resource requirement vector of the VM is complementary to the resource utilization vector of the PM.

Norm-based Greedy ($\mathcal{L}(2)$) For each newly activated PM, the norm-based greedy heuristic algorithm iteratively selects one VM V_i among all unassigned VMs that does not violate the PM's capacity constraint and minimizes the weighted norm distance $\sum_i w_i \|R_i - h_i(t)\|$ between the VM resource request vector R_i and the residual capacity vector of the current PM $H(t) = [h_1(t), \cdots, h_d(t)]$ as well, until no VM can be packed into the current PM.

5.3.3 Comparison

The comparison study among these heuristic algorithms has been performed in [78]. According to [78], *FFDSum* dominates other single dimensional heuristics and dimension aware heuristics can achieve up to 10% improvement over *FFDSum* with realistic

Algorithm 2 DRR-FFD Heuristic

```
1: Cluster VMs by their dominant resources $\{G_1, \cdots, G_d\}$.
2: In each cluster, sort VMs in non-increasing size.
3: Sort PMs in the non-increasing order of the power efficiency.
4: Activate the most power efficient PM and assign the largest VM to it.
5: while not All VMs have been allocated do
6:     Calculate the total residual capacity $\{\sum_j y_j h_{j1}, \cdots, \sum_j y_j h_{jd}\}$ and identify the dominant residual capacity vector $\{D_1^R, \cdots, D_d^R\}$.
7:     Allocate the largest unassigned VM in the non-increasing order of the identified dominant residual capacity vector $\{D_1^R, \cdots, D_d^R\}$ with FFD heuristic.
8: end while
```

workloads, and more consolidation gain can be obtained when VMs are mixed with different dominant resources, such as mix of CPU-intensive VMs and memory-intensive VMs. The heuristic comparison study [78] also shows that if all VMs are constrained by a single resource, the single dimensional heuristics are efficient and can provide almost the same results as dimension-aware heuristics.

5.4 Heterogeneity Aware Dominant Resource Assistant Heuristics

All of the heuristic algorithms discussed in the last section do not consider the heterogeneity of VMs' requirements for different resources. Inspired by the concept of "dominant resource" introduced in [79], which is the resource corresponding to the maximum among all requested resources of a user, several heterogeneity aware dominant resource assistant heuristic algorithms are proposed in this section.

5.4.1 Dominant Residual Resource Aware FFD (DRR-FFD)

Inspired by the efficiency and VM consolidation performance of FFD heuristics, a new FFD derivative heuristic by considering the dominant residual resource, denoted as Dominant Residual Resource aware FFD (DRR-FFD), is proposed in this paper. The basic idea behind

DRR-FFD is that it always tries to balance the resource utilizations across all dimensions by matching up the PM's residual resource capacity with the next VM to be allocated. For example, if more CPU resource than the memory resource is left, a CPU-intensive unassigned VM will be allocated to consume more CPU resource than memory resource. The DRR-FFD heuristic is summarized in Algorithm 2.

Unlike FFD, DRR-FFD clusters the VMs based on their dominant resources, which are the resources corresponding to the largest requirements in the VMs' normalized d-dimensional resource request vectors (line 1 in Algorithm 2). In each cluster, the VMs are sorted in non-increasing order of the predefined VM "size" (line 2 in Algorithm 2). DRR-FFD also sorts PMs in non-increasing order of their power efficiency to ensure that VMs are always assigned to the subset of the most power efficient servers to reduce power consumption (line 3 in Algorithm 2). Similar to FFD, DRR-FFD starts from the largest VM (line 4 in Algorithm 2) and iteratively assigns one VM to the first PM which has enough residual resource capacity to host it until all the VMs have been allocated (lines 5-8 in Algorithm 2).

The major difference between FFD and DRR-FFD is that FFD sorts VMs in a single queue according to their sizes while DRR-FFD organizes VMs into multiple groups $\{G_1, \cdots, G_d\}$ based on the dominant resources. It is simple to determine the next VM to be assigned in FFD since all VMs are organized in a single queue, but DRR-FFD needs to provide a mechanism to determine the next unassigned VM to be allocated. DRR-FFD calculates the total normalized residual resource capacities of all active PMs across all dimensions and identifies the dominant residual resource vector $\{D_1^R, \cdots, D_d^R\}$, which organizes the resources corresponding to the non-increasing order of their residual resource capacities. For example, the total normalized residual resource capacity of all active PMs is [0.2, 0.6] in CPU and memory, respectively. Hence, the dominant residual resource vector is [memory, CPU]. Then, DRR-FFD selects the largest unassigned VM in the cluster which has the largest residual capacity. If there is no unassigned VM in this cluster, DRR-FFD will search for the cluster with the next largest residual capacity. DRR-FFD might run this step on several clusters until it gets a VM to be allocated. Also, as in the last example, DRR-FFD tries the cluster which contains all the memory-intensive VMs first. If this cluster is empty, it tries the cluster which contains all the CPU-intensive VMs.

5.4.2 Individual DRR-FFD (iDRR-FFD)

One problem with DRR-FFD is that the sum of the residual capacities across all active PMs might mislead the actual dominant residual resource. Consider the case with five active PMs. Four of them have the normalized residual resource capacity [0.15 CPU, 0 memory], and the residual resource capacity of the fifth PM is [0.3 CPU, 0.5 memory]. Thus, the total residual resource capacity of these five PMs is [0.9 CPU, 0.5 memory], in which case DRR-FFD will select a CPU-intensive VM to be allocated. However, among these normalized CPU residual capacity, 67% cannot be assigned to any VM because there is no enough memory to satisfy the resource capacity constraint. Also note that the utilization of the CPU resource is already higher than that of memory on the fifth PM. The allocation of another CPU-intensive VM to the fifth PM will even worsen the imbalance of resource utilization. One solution to this problem is to use the residual capacity of each individual PM to identify the dominant residual resource vector. This heuristic is referred to as individual DRR-FFD (iDRR-FFD).

5.4.3 Dominant Residual Resource Based Bin Fill (DRR-BinFill)

Dominant Residual Resource based Bin Fill (DRR-BinFill) heuristic is quite similar to DRR-FFD, except that it focuses on a given server. The DRR-BinFill heuristic is summarized in Algorithm 3. First, DRR-BinFill clusters VMs according to their dominant resources and sorts them in the non-increasing size (lines 1 and 2 in Algorithm 3), and also sorts PMs in the non-increasing order of their energy efficiency (line 3 in Algorithm 3), which are the same as Steps 1-3 of DRR-FFD, and therefore, the idle PMs can be activated in non-increasing order of their energy efficiency, thus ensuring that VMs are allocated to the subset of the most energy-efficient PMs to reduce power consumption (line

Algorithm 3 Dominant Residual Resource based Bin Fill

```
 1: Cluster VMs by their dominant resources $\{G_1, \cdots, G_d\}$.
 2: In each cluster, sort VMs in non-increasing size.
 3: Sort PMs in the non-increasing order of the power efficiency.
 4: while not All VMs has been allocated do
 5:     Activate the most energy efficient idle server $M_i$.
 6:     Allocate the largest unassigned VM to $M_i$.
 7:     Comment: The following is the Bin Filling Process.
 8:     while not Filling process finished do
 9:         Calculate the given server's residual capacity $h_j$ ($j \in [1, d]$) and identify the
            dominant residual capacity vector $\{D_1^R, \cdots, D_d^R\}$.
10:         if The residual capacity of one resource is smaller than $\epsilon$ then
11:             break; //bin filling process finishes.
12:         end if
13:         Search for the clusters corresponding to the order of dominant residual capacity
            vector $\{D_1^R, \cdots, D_d^R\}$ to get a VM to be allocated to the PM $M_i$ without violating
            the resource capacity constraints.
14:         if Such a VM cannot be found then
15:             break; //bin filling process finishes.
16:         end if
17:     end while
18: end while
```

5 in Algorithm 3). The newly activated server is not necessary to be empty, meaning that no VM has been assigned to it. If the given server is empty, the largest unassigned VM will be allocated to it (line 6 in Algorithm 3); otherwise, this step will be skipped. Then, DRR-BinFill will iteratively allocate one VM among all the unassigned VMs to the given server to compensate for the server's residual resource capacity. This procedure is called the bin filling process. In each iteration, DRR-BinFill calculates the given server's residual capacity and identifies the dominant residual capacity vector (line 9 in Algorithm 3), and searches for the VM clusters in the order of the server's dominant residual resource vector (line 13 in Algorithm 3), implying that the larger the given server's residual resource capacity, the earlier the corresponding VM cluster will be searched. The VM searching procedure will be stopped when one VM has been found without violating the given server's capacity constraints. If all clusters have been searched and such a VM cannot be found (lines 14-16 in Algorithm 3), or some of resources in the server have reached their capacity constraints (10-12), the bin filling process will be stopped.

5.5 Performance Evaluation

The performance of the proposed heterogeneity-aware heuristics is evaluated under different synthetic workloads, and the results are compared with other single dimensional heuristics and dimension aware heuristics.

5.5.1 Synthetic Workload

Currently, the commercial VM traces are not available in the public domain due to privacy and security concerns and several synthetic workloads are generated to emulate VM demands and evaluate the proposed heuristics on these synthetic workloads. Most of these synthetic workloads are chosen from the ten classes used by Caprara and Toth [80] to benchmark the effectiveness of two-dimensional bin-packing heuristics. In the first VM class denoted by $Random[h, \alpha, \beta]$, the VM resource request in each dimension is drawn randomly and independently from the range $[\alpha, \beta]$, and the resource capacity of a PM in each dimension is h. In this class $Random[h, \alpha, \beta]$, the dimensions are independently generated and the parameters are set to $h = 150$ and $[\alpha, \beta] = [20, 100]$ as in the experiments performed in [78]. The dimensions in the other two classes, namely, positive correlation $PosCorr[h, \alpha, \beta]$ and negative correction $NegCorr[h, \alpha, \beta]$, are correlated. In both classes, PMs have $h = 150$ capacity in each resource dimension, and for each VM, the demand in the first dimension r_1 is sampled uniformly in the range [20, 100], while the demand in the second dimension r_2 is sampled uniformly in the range $[r_1 - 10, r_1 + 10]$ for the positive correlation class $PosCorr$ and in the range $[110 - r_1, 130 - r_1]$ for the negative correlation class $NegCorr$. The fourth class $HeterExtrCase$ is generated artificially to emulate the worst case of VM consolidation in terms of the heterogeneity of the VMs'requirements for different resources. In this class, the PM capacity is set to $h = 100$ in each resource dimension, and VMs have two types of resource requests. One is the first resource dimension intensive VM request [20,1] and the other is the second resource dimension intensive VM request $[3, 20]$. Each type of VM request has 1000 VMs.

5.5.2 VM Heterogeneity

The effects of the heterogeneity of VMs' requirements for different resources to the consolidation performance of different heuristics are investigated. The ratio of the VM's demands for two different resources $\rho_i^{jk} = r_{ij}/r_{ik}$ $(j \neq k; j, k \in [1, d])$ can reflect the difference of the VM's demands for these two resources. Under the two-dimensional resource requirements, the heterogeneity of VMs' resource demands can be captured with the variance of the ratio of VM's demands for these two resources $\rho_i = r_{i1}/r_{i2}$. The VMs' resource demands are limited in two dimensions in this investigation. Assume the VM resource request in each dimension is drawn randomly and independently from the range $[1, 20]$. Hence, the ratio of the VM's demands for the two resources is in the range $[0.05, 20]$. Further assume the average of the resource demand ratio of all VMs is $\mu \in [0.05, 20]$. The ratio of VM's demands for the two resources is assumed to be a random variable with the normal distribution $\mathcal{N}(\mu, \sigma)$. The demand of the VM's first resource r_{i1} is chosen

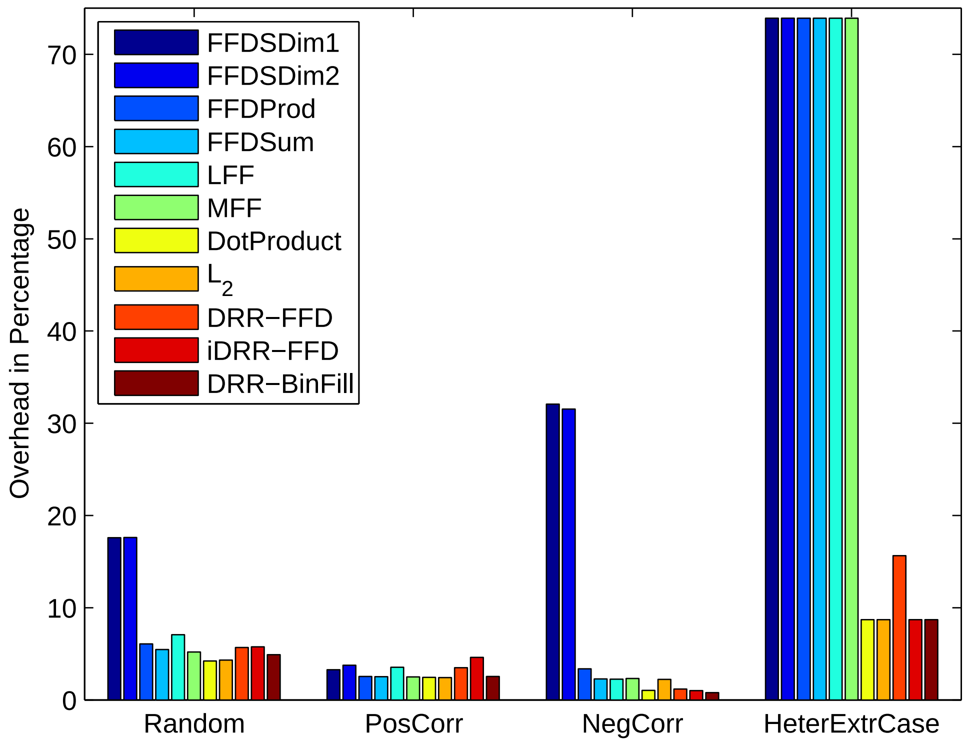


Figure 5.1 VM consolidation performance of different heuristics.

randomly and uniformly in the range $[1, 20]$, and the demand of the VM's second resource is obtained with $r_{i2} = r_{i1}/\rho_i$. If r_{i2} is in the range $[1, 20]$, the VM's resource demands $R_i = \{r_{i1}, r_{i2}\}$ will be kept; otherwise, the demand of the VM's first resource r_{i1} will be reselected uniformly in the range $[1, 20]$, and the demand for the VM's second resource r_{i2} will be recalculated accordingly.

5.5.3 Results

The comparison study among different heuristic algorithms for VM consolidation is performed first with the four classes of synthetic workloads described above and the VM consolidation results are shown in Figure 5.1 in terms of the percentage overhead of the number of active PMs calculated with different heuristic algorithms with respect to the lower bound B_L of the active PMs under different workload scenarios. The lower bound of the number of PMs is estimated with the maximum number of PMs required to satisfy the total resource requests along any single resource dimension $\lceil\sum_{k=1}^{K} r_{ki}/C_i\rceil$ ($i \in [1, d]$) as follows:

$$B_L = max\{\lceil\sum_{k=1}^{K} r_{k1}/C_1\rceil, \cdots, \lceil\sum_{k=1}^{K} r_{kd}/C_d\rceil\} \tag{5.6}$$

From the results shown in Figure 5.1, it can be observed that the consolidation performances of FFDProd, FFDSum, LFF, and MFF are very close to each other. All of these four single dimensional heuristics perform better than another single dimensional heuristic algorithm FFDSDim, and the consolidation results in which VMs are sorted according to the first resource dimension and the second resource dimension are represented as “FFDSDim1” and “FFDSDim2”, respectively. FFDSum performs a little bit better than FFDProd does across all of the four classes of synthetic workloads, implying that the weighted sum of the normalized resource demands is a better metric in sorting VMs in “size” than the product of the normalized resource requirement elements. Therefore, the weighted sum of the normalized resource requirement elements will be used as the metric to sort VMs in other heuristics by default in this chapter, unless otherwise stated. The consolidation performance of LFF is worse than that of MFF, because LFF also takes care of the load balance among the active PMs except VM consolidation. The dominant aware heuristics, DotProduct and Norm-based greedy $\mathcal{L}_2$, can achieve better consolidation performance than those of single dimensional heuristics, especially in the fourth synthetic workload scenario $HeterExtrCase$. Note that the variance of the ratios of the VM’s requirements for two different resources is much larger than that in other three scenarios. The performances of DotProduct and Norm-based greedy $\mathcal{L}_2$ are nearly the same to each other. It is worth observing that the proposed heterogeneity aware dominant resource assistant heuristics, namely DRR-FFD, iDRR-FFD and DRR-BinFill, perform quite well in all of these four scenarios.

The performance of different heuristics under different VM heterogeneity in terms of the standard deviation of the resource ratios $\rho_i = r_{i1}/r_{i2}$ is shown in Figure 5.2. This

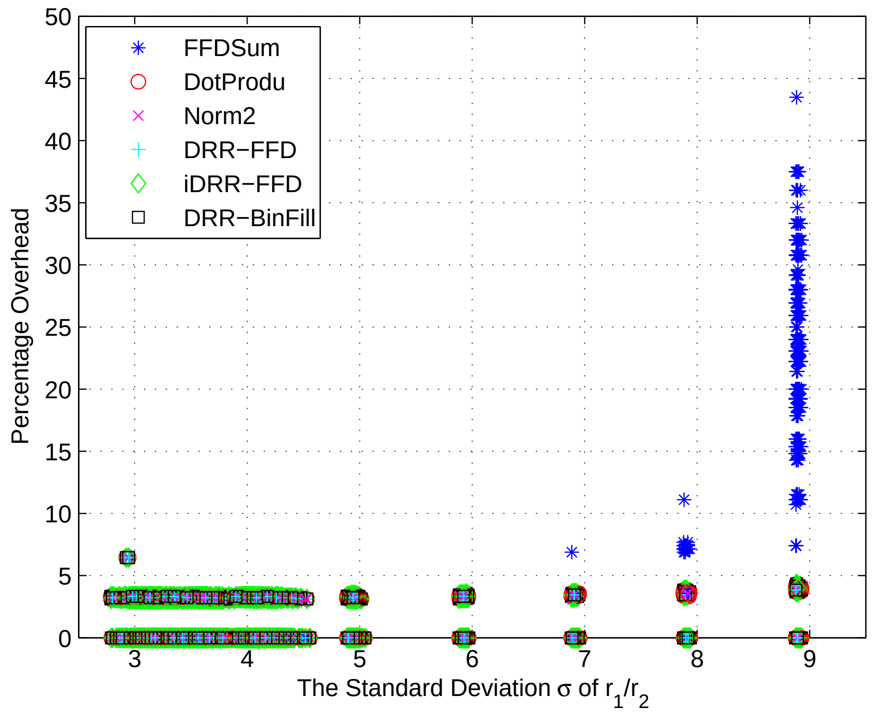


Figure 5.2 VM consolidation performance over different resource requirement heterogeneity.

figure shows clearly that FFDSum performs poorly with large resource requirement ratios. Both the dimension aware heuristics, DotProduct and Norm-based greedy $\mathcal{L}_2$, and the heterogeneity aware dominant resource assistant heuristics, DRR-FFD, iDRR-FFD, and DRR-BinFill, perform quite well with large resource requirement ratios. As compared to the dimension aware heuristics, the proposed heterogeneity aware heuristics save the computations in calculating the dot product between the residual capacity vector of a PM and the VM resource request vector, or the weighted norm distance between these two vectors.

5.6 Summary

In this chapter, several heterogeneity-aware dominant resource assistant heuristic algorithms have been proposed for virtual machine consolidation by exploring the heterogeneity of the virtual machine's requirements for different resources and utilizing the dominant resources to assist VM consolidation. The performance of the proposed heuristic algorithms has been evaluated with different classes of synthetic workloads under different VM requirement heterogeneity conditions. The simulation results have shown that the proposed heterogeneity-aware dominant resource assistant heuristics achieve better consolidation performance than single dimensional heuristics, i.e., FFDProd and FFDSum, with a little extra effort in clustering VMs according to their dominant resources and identifying the dominant residual resources. The results also show that the proposed heterogeneity-aware heuristics can achieve quite similar consolidation performance as dimension-aware heuristics with reduced computation.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Data centers, facilities with communications network equipments and servers for data processing and/or storage, are prevalent and essential to host a myriad of diverse services and applications for various private, non-profit, and government systems. With rapid deployment of modern high-speed low-latency large-scale data centers, many problems have been observed in data centers. In this dissertation, the Ethernet layer congestion control, unified congestion control, power consumption and VM consolidation issues in data centers have been studied.

6.1 Contributions

An enhanced QCN congestion notification algorithm, called Fair QCN (FQCN), has been proposed to improve fairness allocation of bottleneck link capacity among all sharing sources. FQCN identifies congestion culprits through joint queue and per flow monitoring, feedbacks individual congestion information to each identified congestion culprit through multi-casting, and ensures convergence to statistical fairness. The stability and fairness of FQCN is analyzed by using Lyapunov functions and demonstrated that FQCN is stable and closer to max-min fairness than to proportional fairness. The performance of FQCN has been evaluated with different traffic dynamics in terms of the queue length stability, link throughput and rate allocations to traffic flows with different traffic dynamics by NS2 simulations under three network topologies, namely, the dumb-bell topology, parking-lot topology and a simple and representative TCP Incast network topology, and the effects on TCP Incast has also been investigated. The simulation results have shown that FQCN can successfully allocate the fair share rate to each traffic source while maintaining the queue length stability. As compared to AF-QCN, the flow sending rates with FQCN are more stable with respect to the fair share rates. FQCN can also provide weighted fair share allocation of the bottleneck link capacity. In addition, FQCN significantly enhances TCP throughput performance with respect to TCP Incast, and achieves better TCP throughput than QCN and AF-QCN in a TCP Incast setting.

Furthermore, a unified congestion detection, notification and control system for data center networks has been designed to efficiently resolve network congestion in a uniform solution and to ensure convergence to statistical fairness with "no state" switches simultaneously. The proposed congestion control framework decouples the congestion control intelligence from switches, and thus simplifies the implementation and upgrade of the congestion control algorithms. Also, it incorporates a distributed potential congestion culprit detection mechanism and does not require stateful switches, i.e., switches monitor the local congestion status, forward the congestion status to the centralized congestion controller, and do not need to store per-flow information. The proposed congestion control system can easily be incorporated with any congestion notification and control algorithm, regardless whether per-flow information is required or not. Besides, it can be deployed for cross-layer congestion control. The FQCN algorithm has been incorporated in the proposed system, and its performance has been evaluated through extensive simulations. The simulation results validate the successful allocation of fair share rates and the maintenance of queue length stability of the proposed congestion control system.

Also, a **h**ierarchical **e**ne**r**gy **o**ptimization (HERO) model has been proposed and evaluated to reduce the power consumption of data center networks without violating the connectivity and Quality of Service (QoS) constraints. Especially, the power optimization of networking elements in a DCN by switching off the idle switches or links can be divided into core-level and pod-level network elements power optimization. Given a physical DCN topology, the traffic matrix, and the power consumption of each link and switch, the designed core-level and pod-level power optimization problems in the proposed HERO model fall in the class of capacitated multi-commodity minimum cost flow (CMCF) problem, which is NP-hard. Several heuristic algorithms have been designed based on different switch elimination criteria to solve the hierarchical optimization problem for large data centers. The switch elimination criterion could be switch throughput, switch power consumption, and switch power efficiency. The basic idea of the proposed heuristic algorithms is to try to turn off the core switches and core-level links iteratively as well as the aggregation switches and pod-level links iteratively in the core-level and pod-level power optimization, respectively. Since the commercial data center traffic traces were not available in the public domain due to privacy and security concerns, the performance of the proposed HERO model has been evaluated by generating several traffic patterns to emulate typical data center workloads, including random elephant, all-to-all data shuffle, staggered (P_{out}, P_{edge}, P_{pod}) and diurnal traffic variation. The simulation results show that the proposed HERO model can achieve optimized power consumptions of a DCN similar to that of the non-hierarchical model but with much computational efforts. The algorithm complexities of HERO and non-hierarchical energy optimization problems have been compared in terms of the total number of variables and the total number of constraints in a fat-tree DCN. As compared to the non-hierarchical energy optimization model, at least 65% and 60% of variables and constraints can be reduced with the HERO model, respectively. The simulation results show that among all of the proposed heuristic algorithms, switch power consumption heuristic algorithm based on the total power consumption of the core and aggregation switches in the core-level power optimization achieves the minimum power consumption since it considers the power consumption of core switches and aggregation switches in a whole instead of separately.

Finally, several heterogeneity aware dominant resource assistant heuristic algorithms, namely, dominant residual resource aware first-fit decreasing (DRR-FFD), individual DRR-FFD (iDRR-FFD) and dominant residual resource based bin fill (DRR-BinFill), for VM consolidation have been proposed. The proposed virtual machine consolidation heuristic algorithms explore the heterogeneity of the VMs' requirements for different resources as well as the heterogeneity of the PMs' resource capacities and utilize the dominant resource (i.e., the resource corresponding to the largest element in the vector of the VM's resource demand or the PM's resource capacity) to assist VM consolidation. The performance of the proposed heuristic algorithms has been evaluated with different classes of synthetic workloads under different VM requirement heterogeneity conditions. The simulation results have validated that the proposed heterogeneity-aware dominant resource assistant heuristics achieve quite similar consolidation performance as dimension-aware heuristics with almost the same computational cost as those of the single dimensional heuristics.

6.2 Future Work

Current state-of-the-art congestion control algorithms cannot provide a unified and efficient congestion control algorithm for data center networks. Based on the proposed congestion control framework in this thesis, a suit of congestion control protocols will be designed to enable the network to dynamically detect network congestion, estimate and collect traffic flow information, and finally resolve network congestion efficiently. Requirements on the designed cross-layer multi-optional congestion control protocols include:

1. The designed protocols must be efficient to resolve network congestion. They must be able to detect and react to network congestion quickly.
2. The designed protocols must be robust to congestion message losses and delays, including the transmission delays and congestion message processing delays.
3. The designed protocols should minimize the congestion message overhead and maximize the utilization of the congestion information.
4. The detected congestion information can be utilized by multiple layer protocols.

Moreover, the interactions of congestion control algorithms deployed on different layers have not been investigated properly in the previous works. In this part, future works will focus on addressing the following issues:

1. When multiple congestion control mechanisms are deployed in a single data center, the interactions among different congestion control algorithms should be investigated.

2. The influence of current state-of-the-art congestion control algorithms on other layer protocols should be studied. For example, Ethernet layer congestion control mechanisms relieve network congestion by regulating the Ethernet transmission rate. The influence of Ethernet layer congestion control algorithms on TCP throughput and traffic routing should be investigated.

3. When network congestion happens, Ethernet layer, transport layer, or network layer congestion control mechanism or multiple congestion control mechanisms should be executed to resolve network congestion. Which layer of the congestion control algorithms is the most effective solution to relieve network congestion?

4. The root cause to network congestion could be quite different from each other. For example, some network congestion results from the interactions of multiple elephant flows, while other network congestion is caused by some aggressive traffic flows. Determining the root causes of network congestion and the influence of different root causes to network congestion on the efficiency of different congestion control algorithms will be a challenging endeavor.

Machine virtualization is one of the mostly used techniques to provide flexible and cost-effective resource sharing among users in data centers. Consolidating virtual machines (VMs) to physical servers in data center infrastructure helps to increase the utilization of data center resources, improve the energy efficiency of data centers since the idle infrastructures can be shut down or put into low power mode to save energy, balance the workload across the whole data center to avoid hotspots in data centers, and ensure the quality of service for cloud computing applications. Joint consideration of these conflicting objectives to investigate the pairwise tradeoff is still a challenging problem to virtual machine placement, and this multi-objective joint virtual machine placement and traffic engineering problem will be studied in the future.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google File System,” *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[2] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: A Distributed Storage System for Structured Data,” *ACM Transactions on Computer Systems*, vol. 26, no. 2, pp. 1–26, 2008.

[3] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] D. Bergamasco, “Data Center Ethernet Congestion Management: Backward Congestion Notification,” in *Proc. of IEEE 802.1 Meeting*, Berlin, Germany, May 2005.

[5] J. Jiang, R. Jain, and C. So-In, “An Explicit Rate Control Framework for Lossless Etherent Operation,” in *Proc. of the International Conference on Communications (ICC)*, Beijing, China, May 19-23, 2008, pp. 5914–5918.

[6] C. So-In, R. Jain, and J. Jiang, “Enhanced Forward Explicit Congestion Notification (E-FECN) Scheme for Datacenter Ethernet Networks,” in *Proc. of International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Edinburgh, UK, Jun. 2008, pp. 542–546.

[7] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, “Data Center Transport Mechanisms: Congestion Control Theory and IEEE Standardization,” in *Proc. of the 46th Annual Allerton Conference*, Illinois, USA, Sep. 2008, pp. 1270–1277.

[8] QCN Pseudo-code Version 2.0, http://www.ieee802.org/1/files/public/docs2008/au-rong-qcn-serial-hai-pseudo-code%20rev2.0.pdf (accessed on Feb. 10, 2014).

[9] M. Alizadeh, A. Kabbani, B. Atikoglu, and B. Prabhakar, “Stability Analysis of QCN: The Averaging Principle,” in *Proc. of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, San Jose, CA, June 7-11, 2011.

[10] J. Jiang and R. Jain, “Analysis of Backward Congestion Notification (BCN) for Ethernet In Datacenter Applications,” in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Alaska, USA, May 6-12, 2007, pp. 2456–2460.

[11] APC White Paper 6, “Determining Total Cost of Ownership for Data Center and Network Room Infrastructure,” http://www.linuxlabs.com/PDF/Data%20Center%20Cost%20of%20Ownership.pdf (accessed on Feb. 20, 2011).

[12] L. A. Barroso and U. Hölzle, “The Case for Energy-Proportional Computing,” *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[13] Data Center Bridging Task Group, http://www.ieee802.org/1/pages/dcbridges.html (accessed on Feb. 25, 2014).

[14] P. Devkota and A. Reddy, "Performance of Quantized Congestion Notification in TCP Incast Scenarios of Data Centers," in *Proc. of IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, Miami Beach, Florida, Aug. 17-19, 2010, pp. 235–243.

[15] Y. Zhang and N. Ansari, "On Mitigating TCP Incast in Data Center Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Shanghai, China, April 2011, pp. 51–55.

[16] ——, "Fair Quantized Congestion Notification in Data Center Networks," *IEEE Transactions on Communications*, vol. 61, no. 11, pp. 4690–4699, November 2013.

[17] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar, "AF-QCN: Approximate Fairness with Quantized Congestion Notification for Multi-tenanted Data Centers," in *Proc. of IEEE the 18th Annual Symposium on High Performance Interconnects (HOTI)*, Mountain View, California, USA, Aug. 18-20, 2010, pp. 58–65.

[18] Y. Zhang and N. Ansari, "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 39–64, 2013.

[19] J.-Y. Boudec, *Rate Adaptation, Congestion Control and Fairness: A Tutorial.* Ecole Polytechnique Fédérale de Lausanne (EPFL), Dec. 2000.

[20] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, March 1998.

[21] S. Golestani and S. Bhattacharyya, "A Class of End-to-End Congestion Control Algorithms for the Internet," in *Proc. of the Sixth International Conference on Network Protocols, 1998*, Austin, Texas, USA, Oct. 13-16, 1998, pp. 137–150.

[22] NS-2 Network Simulator, http://www.isi.edu/nsnam/ns/ (accessed on Feb. 25, 2014).

[23] D. Nagle, D. Serenyi, and A. Matthews, "The Panasas ActiveScale Storage Cluster: Delivering Scalable High Bandwidth Storage," in *Proc. of the ACM/IEEE conference on Supercomputing*, Washington, DC, Nov. 6-12, 2004.

[24] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan, "Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems," in *Proc. of the 6th USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, Feb. 26-29, 2008, pp. 1–14.

[25] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," in *Proc. of the 1st ACM workshop on Research on enterprise networking (WREN)*, Barcelona, Spain, August 21, 2009, pp. 73–82.

[26] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication," in *Proc. of ACM SIGCOMM*, Barcelona, Spain, Aug. 17-21, 2009, pp. 303–314.

[27] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine Grained Traffic Engineering for Data Centers," in *Proc.of the Seventh Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Tokyo, Japan, Dec. 6-9, 2011, pp. 8:1–8:12.

[28] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," in *Proc. of the 7th USENIX conference on Networked Systems Design and Implementation (NSDI)*, San Jose, California, April 28-30, 2010.

[29] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-Overhead Datacenter Traffic Management using End-Host-Based Elephant Detection," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Shanghai, China, April 10-15, 2011, pp. 1629–1637.

[30] X. Wu and X. Yang, "DARD: Distributed Adaptive Routing for Datacenter Networks," in *IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*, Macau, China, June 18-21, 2012, pp. 32–41.

[31] 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 ACM SIGCOMM*, New Delhi, India, Aug. 30-Sept. 3, 2010, pp. 63–74.

[32] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast Congestion Control for TCP in Data Center Networks," in *Proc. of the 6th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Philadelphia, Pennsylvania, 2010, pp. 13:1–13:12.

[33] Software Defined Networking, http://en.wikipedia.org/wiki/Software-defined_networking (accessed on Feb. 25, 2014).

[34] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The Nature of Data Center Traffic: Measurements & Analysis," in *Proc. of the 9th ACM SIGCOMM conference on Internet measurement conference (IMC)*, Chicago, Illinois, USA, Nov.4-6, 2009, pp. 202–208.

[35] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding Data Center Traffic Characteristics," in *Proc. of ACM workshop on Research on Enterprise Networking*, Barcelona, Spain, Aug. 2009, pp. 65–72.

[36] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proc. of Internet Measurement Conference (IMC)*, Melbourne, Australia, Nov. 1-3, 2010, pp. 267–280.

[37] U.S. Environmental Protection Agency, "Environmental Protection Agency, Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431," *ENERGY STAR Program*, Aug. 2007.

[38] J. G. Koomey, "Growth in Data Center Electricity Use 2005 to 2010," August 4, 2011.

[39] R. Sharma, A. Shah, C. Bash, T. Christian, and C. Patel, "Water Efficiency Management in Datacenters: Metrics and Methodology," in *Proc. of IEEE International Symposium on Sustainable Systems and Technology (ISSST)*, Phoenix, AZ, May 18-20, 2009.

[40] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee and N. McKeown, "ElasticTree: Saving Energy in Data Center Networks," in *Proc. of the 7th Symposium on Networked Systems Design and Implementation*, San Jose, CA, Apr. 2010, pp. 249–264.

[41] S. Dawson-Haggerty, A. Krioukov, and D. Culler, "Power Optimization - a Reality Check," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-140, Oct. 2009.

[42] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power Awareness in Network Design and Routing," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Phoenix, AZ, USA, April 13-18, 2008, pp. 457–465.

[43] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A Power Benchmarking Framework for Network Devices," in *Proc. of the 8th International IFIP-TC 6 Networking Conference*, Aachen, Germany, May 11-15, 2009, pp. 795–808.

[44] ——, "A Power Benchmarking Framework for Network Devices," in *Proc. of the 8th International IFIP-TC 6 Networking Conference*, Aachen, Germany, May 11-15, 2009, pp. 795–808.

[45] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," in *Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, California, April 16-18, 2008, pp. 337–350.

[46] S. Srikantaiah, A. Kansal and F. Zhao, "Energy Aware Consolidation for Cloud Computing," in *Proc. of Conference on Power Aware Computing and Systems*, San Diego, CA, Dec. 2008.

[47] A. Verma, R. Koller, L. Useche, and R. Rangaswami, "SRCMap: Energy Proportional Storage Using Dynamic Consolidation," in *Proc. of the 8th USENIX conference on File and storage technologies (FAST)*, San Jose, California, Feb. 23-26, 2010.

[48] Fujitsu Siemens Computers, "Virtualization and the Green Data Center."

[49] C. Bash and G. Forman, "Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center," HP Laboratories, Tech. Rep. HPL-2007-62, 2007.

[50] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, “Making Scheduling Cool: Temperature-Aware Workload Placement in Data Centers,” in *Proc. of the Annual Conference on USENIX Annual Technical Conference (ATEC)*, Anaheim, CA, April 10-15, 2005, pp. 61–75.

[51] F. Ahmad and T. N. Vijaykumar, “Joint Optimization of Idle and Cooling Power in Data Centers While Maintaining Response Time,” in *Proc. of the fifteenth edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Pittsburgh, Pennsylvania, USA, 2010, pp. 243–256.

[52] Y. Shang, D. Li, and M. Xu, “Energy-Aware Routing in Data Center Network,” in *Proc. of ACM SIGCOMM*, New Delhi, India, Aug. 30-Sep.3 2010, pp. 1–8.

[53] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, “VMFlow: Leveraging VM Mobility to Reduce Network Power Costs in Data Centers,” in *Proc. of the 10th International IFIP TC six conference on Networking*, Valencia, Spain, May 9-13, 2011, pp. 198–211.

[54] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, “VMPlanner: Optimizing Virtual Machine Placement and Traffic Flow Routing to Reduce Network Power Costs in Cloud Data Centers,” *Computer Networks*, vol. 57, no. 1, pp. 179–196, Jan. 2013.

[55] Cisco Data Center Infrastructure 2.5 Design Guide, http://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmigration_09186a008073377d.pdf (accessed on Feb. 25, 2014).

[56] 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,” in *Proc. of ACM SIGCOMM*, Barcelona, Spain, August 17-21, 2009, pp. 51–62.

[57] 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 *Proc. of ACM SIGCOMM*, Barcelona, Spain, August 17-21, 2009, pp. 39–50.

[58] 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 *Proc. of ACM SIGCOMM*, Seattle, WA, August 17-21, 2008, pp. 75–86.

[59] 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 *Proc. of ACM SIGCOMM*, Barcelona, Spain, August 17-21, 2009, pp. 63–74.

[60] C. E. Leiserson, “Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing,” *IEEE Trans. on Computer*, vol. 34, no. 10, pp. 892–901, Oct. 1985.

[61] I. Ghamlouche, T. G. Crainic, and M. Gendreau, “Cycle-Based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design,” *Operations Research*, vol. 51, no. 4, pp. 655–667, 2003.

[62] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing Power Consumption in Backbone Networks," in *Proc. of IEEE International Conference on Communications (ICC)*, Dresden, Germany, Jun.14-18, 2009, pp. 1–6.

[63] L. Chiaraviglio, M. Mellia, and F. Neri, "Energy-Aware Backbone Networks: A Case Study," in *Proc. of IEEE International Conference on Communications*, Dresden, Germany, Jun.14-18, 2009, pp. 1–5.

[64] IBM ILOG CPLEX Optimizer, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (accessed on Feb. 25, 2014).

[65] D. S. Hochbaum, *Approximation Algorithms for NP-hard Problems*. Boston, MA, USA: PWS Publishing Co., 1997.

[66] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *Proc. of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware)*, Leuven, Belgium, Dec. 1-5, 2008, pp. 243–264.

[67] A. Beloglazov and R. Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in *Proc. of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID)*, Melbourne, Australia, May 17-20, 2010, pp. 826–831.

[68] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, May 2012.

[69] Y. Wang and X. Wang, "Power Optimization with Performance Assurance for Multi-tier Applications in Virtualized Data Centers," in *Proc. of the 39th International Conference on Parallel Processing Workshops (ICPPW)*, San Diego, CA, USA, Sept. 13-16, 2010, pp. 512–519.

[70] N. Ansari and E. Hou, *Computational Intelligence for Optimization*. New York, NY: Springer Publishing Company, Incorporated, 1997.

[71] E. S. H. Hou, N. Ansari, and H. Ren, "A Genetic Algorithm for Multiprocessor Scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 2, pp. 113–120, Feb. 1994.

[72] J. Xu and J. A. B. Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments," in *Proc. of 2010 IEEE/ACM Internaltional Conference on Green Computing and Communications (GreenCom) & 2010 IEEE/ACM Internaltional Conference on Cyber, Physical and Social Computing (CPSCom)*, Hangzhou, China, Dec. 18-20, 2010, pp. 179–188.

[73] H. Hlavacs and T. Treutner, "Genetic Algorithms for Energy Efficient Virtualized Data Centers," in *Proc. of the 8th International Conference on Network and Service Management (CNSM)*, Oct. 2012, pp. 422–429.

[74] G. Wu, M. Tang, Y.-C. Tian, and W. Li, "Energy-Efficient Virtual Machine Placement in Data Centers by Genetic Algorithm," in *Proc. of the 19th international conference on Neural Information Processing - Volume Part III (ICONIP)*, Doha, Qatar, Nov. 12-15, 2012, pp. 315–323.

[75] Y. Wu, M. Tang, and W. Fraser, "A Simulated Annealing Algorithm for Energy Efficient Virtual Machine Placement," in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2012, pp. 1245–1250.

[76] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration," in *Proc. of the 4th USENIX conference on Networked systems design and implementation (NSDI)*, Cambridge, MA, April 11-13, 2007.

[77] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage Virtualization: Integration and Load Balancing in Data Centers," in *Proc. of the ACM/IEEE conference on Supercomputing*, Austin, Texas, Nov. 15-21, 2008, pp. 53:1–53:12.

[78] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, L. Uyeda, and U. Wieder, "Validating Heuristics for Virtual Machines Consolidation," Microsoft, Tech. Rep. MSR-TR-2011-9, Sept. 2011. [Online]. Available: http://research.microsoft.com/pubs/144571/virtualization.pdf

[79] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant Resource Fairness: Fair Allocation of Multiple Resource Types," in *Proc. of the 8th USENIX conference on Networked Systems Design and Implementation (NSDI)*, Boston, MA, March 30-April 1, 2011.

[80] A. Caprara and P. Toth, "Lower Bounds and Algorithms for the 2-dimensional Vector Packing Problem," *Discrete Applied Mathematics*, vol. 111, no. 3, pp. 231–262, Aug. 2001.