DELAY MODELING IN DATA CENTER NETWORKS: A TAXONOMY AND PERFORMANCE ANALYSIS

A thesis submitted
to Kent State University in partial fulfillment of the requirements for the degree of Master of Science

by
Reem Abdullah Alshahrani

August 2013
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................... VII

LIST OF TABLES ............................................................................................................... IX

DEDICATION ..................................................................................................................... X

ACKNOWLEDGEMENTS ................................................................................................. XI

CHAPTER 1 INTRODUCTION ......................................................................................... 1

1.1 Design Principles of DCNs ....................................................................................... 3

1.2 Thesis contribution .................................................................................................... 5

1.3 Thesis Organization ................................................................................................... 5

CHAPTER 2 PREVIOUS AND CURRENT WORK ......................................................... 6

2.1 Data Centers Networking ......................................................................................... 6

2.2 DCNs Classification ................................................................................................. 7

2.3 Hierarchical Structures .......................................................................................... 8

2.3.1 Fat-tree .................................................................................................................. 9

2.3.2 VL2 ......................................................................................................................... 10

2.3.3 Portland ................................................................................................................ 11

2.4 Random Structures ............................................................................................... 12

2.4.1 Jellyfish ................................................................................................................ 13

2.4.2 Small World Data Centers (SWDC) ................................................................... 14

2.4.3 Scafida .................................................................................................................. 15

2.4.4 SPAIN ................................................................................................................... 16
CHAPTER 3 TRANSMISSION CONTROL PROTOCOL (TCP) ISSUES IN DATA CENTER NETWORKS ................................................................. 29

3.1 Introduction .................................................................................................................... 29

3.2 TCP Incast ...................................................................................................................... 30

3.3 Congestion Trees .......................................................................................................... 32

3.4 TCP Outcast ................................................................................................................. 33

3.5 Traffic Patterns within Data Center Networks .............................................................. 33

3.6 Proposed Solutions ....................................................................................................... 36

3.6.1 DCTCP .................................................................................................................. 36

3.6.2 ICTCP .................................................................................................................... 37
3.6.3 MPTCP .................................................................................................................. 37

CHAPTER 4 ROUTING SCHEMES ............................................................................. 39
4.1 Routing in DCNs ...................................................................................................... 39
4.2 Routing Optimization .............................................................................................. 40
4.3 Centralized routing ................................................................................................. 41
  4.3.1 Hedera .................................................................................................................. 41
4.4 Distributed Routing ............................................................................................... 42
  4.4.1 Equal-Cost-Multi-Path (ECMP) ......................................................................... 42
  4.4.2 Valiant Load Balancing (VLB) .......................................................................... 42
  4.4.3 DARD .................................................................................................................... 43
4.5 Load-Balancing Techniques .................................................................................. 43

CHAPTER 6 CASE STUDY: FAT-TREE TOPOLOGY .............................................. 45
6.1 Theoretical Background on Fat-tree ....................................................................... 45
6.2 Approximated Fat-tree ........................................................................................... 46
6.3 Original Fat-tree and Approximated Fat-tree in Communication Network
Environment .................................................................................................................. 47
6.4 Bottlenecks in Fat-tree .......................................................................................... 49
6.5 Utilization in Fat-tree .............................................................................................. 50
6.6 Theoretical Analysis of End-to-End Delay in Fat-tree Topology .............................. 51
  6.6.1 End-to-End Delay Analysis for Downlink Traffic (Client-Server) .................. 54
  6.6.2 End-to-End Delay Analysis for Uplink Traffic (Server-Client) ...................... 58
LIST OF FIGURES

Figure 1. DCNs classification ................................................................. 8
Figure 2. Simple fat-tree topology ............................................................ 9
Figure 3. Jellyfish topology with 16 servers ............................................. 13
Figure 4. Small World Data Center with ring topology .......................... 14
Figure 5. DCell topology ................................................................. 17
Figure 6. BCube with n=4 ............................................................... 19
Figure 7. Illustration of DCNs layers ...................................................... 27
Figure 8. TCP incast in DCNs .............................................................. 31
Figure 9. Global data centers traffic by destination ................................. 35
Figure 10. Original fat-tree topology ......................................................... 46
Figure 11. Approximated fat-tree topology ............................................. 47
Figure 12. Concentration in fat-tree topology ........................................... 49
Figure 13. Abstraction of fat-tree ....................................................... 50
Figure 14. A node configuration with aggregate traffic ............................ 52
Figure 15. Downlink traffic (client -server) in fat-tree ............................. 55
Figure 16. Aggregate arrival flows for downlink traffic ......................... 57
Figure 17. Uplink traffic (server-client) in fat-tree .................................. 59
Figure 18. Aggregate arrival flows for uplink traffic ................................. 60
Figure 19. End-to-end delay comparison for fat-tree ............................... 62
Figure 20. OPNET Modeler simulation cycle. .......................................................... 64
Figure 21. Fat-tree modeling using OPNET ............................................................ 68
Figure 22. Cisco modeling using OPNET ............................................................. 68
Figure 23. Applications in fat-tree DCN .............................................................. 70
Figure 24. Applications in Cisco DCN ................................................................. 70
Figure 25. Average delay for fat-tree and Cisco DCN .......................................... 71
Figure 26. Average throughput for fat-tree and Cisco DCN .................................. 71
Figure 27. Applications’ delay for fat-tree and Cisco DCN ................................... 72
Figure 28. Applications’ throughput for fat-tree and Cisco DCN ......................... 72
LIST OF TABLES

Table 1. DCNs layers. ........................................................................................................ 26

Table 2. End-to-end delay for downlink traffic ................................................................. 58

Table 3. End-to-end delay for uplink traffic. ................................................................. 61

Table 4. Fat-tree simulation parameters........................................................................ 66

Table 5. Cisco DCN simulation parameters. ............................................................... 66

Table 6. Workload settings for applications simulation. .............................................. 69
DEDICATION

I dedicate this thesis to my dear husband, Waleed for his endless support and encouragement. A special gratitude for all you have given me.
ACKNOWLEDGEMENTS

I would like to express my great appreciation to Prof. Hassan Peyravi for his valuable and helpful suggestions during the planning and development of this thesis.

Reem Alshahrani

July 10th, Kent, Ohio
CHAPTER 1

Introduction

In recent years, data centers or servers’ farms have been increasingly used in hospitals, universities and enterprise to run a variety of applications that meet consumers’ needs. These applications include Web services, online social networking, data analysis, computation intensive applications and scientific computing. A data center is a large cluster of computers that is owned and operated by an organization. They are mainly designed with the goal of optimizing both cost and performance. Data centers became the most effective way to accomplish large-scale computational needs with high performance using the smallest amount of resources. Cisco Cloud Index [1] is expecting that in the future a significant portion of Internet applications and communications will take place within Data Center Networks (DCNs). Since computations became constrained by the underlying interconnection networks, DCNs became a crucial design element in system architecture. Therefore, researchers have begun to explore how to better design and manage DCNs.

Data centers require high performance components for their inter-process communication, storage and networking. Given hardware is most significant component of any data center, in recent years, DCNs designers tend to use Commercial off-the-shelf (COTS) components with low cost deployment. Most of DCNs are built with off-the-shelf rack-mount servers interconnected by COTS Gigabit Ethernet switches.
The evolution from 1 Giga-bit to 10 Giga-bit Ethernet and the emerging of 40 Giga-bit Ethernet has exposed performance bottlenecks in the communication stack that requires better hardware-software coordination for efficient communications. As the network bandwidth continues to increase faster than server capacity, data centers are anticipated to be the bottleneck in hosting network-based services. It has been observed that data centers contribute to approximately 40% of overall delay, and this delay is likely to grow with the increasing use of dynamic Web contents [2]. By some estimates, the Internet is growing at a rate of 70-150% annually [1]. The recent growth in Internet traffic is related to some of the new services, such as video/audio streaming, cloud computing and backup services [2]. This drives most businesses to expand their DCNs to keep up with their growth.

The evolving virtualization and consolidation of data centers and their resources require a resilient, highly scalable and secure data network bases. The network provides secure user data access, aggregation and interconnection of shared data center components to meet the required specifications including storage systems, servers’ applications and devices. A well-planned data center networks should be able to ensure data integrity and data availability efficiently. They must also incorporate scalability in terms of cost-effective expansion without performance degradation nor creating bottlenecks.
1.1 Design Principles of DCNs

Since Ethernet commodity switches have been designed for Local Area Networks (LAN), they cannot be easily used efficiently in DCNs. There are some special requirements that should be satisfied in DCNs design to ensure QoS in these networks. Major requirements are as follows:

**Reliability:** Network performance and reliability are key design goals for any DCN. Many DCNs users such as universities, service providers, and enterprises rely on data movement and resources allocation in their DCNs. Reliability is thus an important issue for both providers and users. Developing more efficient routing schemes and exploring routing redundancy can improve robustness of data centers.

**Scaling:** The physical structure of a DCN should be scalable. It must connect hundreds, or thousands of servers efficiently at a minimal cost while maintaining the minimum performance requirements. It should support the incremental logical expansion and physical deployment. Moreover, the protocol designs such as routing algorithm to be scalable.

**Fault tolerance:** Fault tolerance in a DCN makes the network more robust and reliable. Failure can occur due to software, hardware, and power outage problems. Failures in current DCNs are common issues especially in the container sized data centers or modular data centers. Container-sized data centers are portable data centers that can be deployed anywhere to add more capacity to the network as needed. Because once the modular data center sealed by the manufacturer, it cannot be opened again. Thus, fault
tolerance is very important for topologies that were designed for the container-sized data centers such as BCube [5].

**Load balancing:** Using load balancing techniques within a DCN ensure better link capacity utilization with high throughput and low latency. Careful rearrangement of servers with non-blocking interconnections will provide guaranteed performance for divers data center applications with different performance requirements.

**Cost:** Some DCNs use costly switches to address network bottlenecks. They trade network blocking/bottleneck with hardware cost and network underutilization. Research has shown [3] network degradation in terms of utilization or cost over performance. Thus, the cost of building and maintaining a DCN could be significantly high without providing the expected performance. With careful interconnection and switching fabrications, optimal performance, in terms of throughput and delay can be achieved with minimal cost. Cost is not only hardware cost, but algorithmic cost associated with interconnection routing and fault-tolerance.

**High network capacity:** Many online services such as cloud computing need a huge amount of network bandwidth to deliver acceptable run-time performance. Of course, design principles of DCNs are not limited by the mentioned issues, there are more issues need to be considered in the DCN design such as power consumption and simplicity.
The thesis contribution is four-fold. First, a taxonomy and classification of data center networks based on their underlying interconnections has been investigated. Second, the performance of data centers in terms of throughput and delay has been investigated by simulation and theoretical methods. In specific, we use an open queuing system based on Jackson’s theorem to investigate the queuing delay which is a dominant factor in any data center. We focus on the topology of servers’ interconnections which has significant impact on data centers performance. The results of this analysis provide a basis for optimized DCNs by investigating the limitations and bottlenecks in the current DCNs.

This thesis is organized as follows. Chapter 2 includes a literature review on recent data center networks with their classification. In Chapter 3, we investigate TCP issues and the implications of different traffic patterns in data center networks. Chapter 4 introduces a theoretical analysis of fat-tree followed by routing schemes and routing optimization. Chapter 5 covers simulations of data centers followed by their performance results. Finally, conclusions, remarks and future work are given in Chapter 6.
CHAPTER 2

Previous and Current Work

2.1 Data Centers Networking

Large-scale DCNs are the foundations to support many data-intensive communications such as scientific computations, Internet applications, and data analysis. A DCN is a house of tens of thousands of servers that dynamically share processors, memory, and I/O devices. These servers are packaged into racks and connected with an aggregation Top of Rack (ToR) switches. ToR switches connect all racks using COTS Ethernet switches. These servers are organized as pods or clusters that are tightly connected with a high-bandwidth low latency network. However, the bandwidth at the top-level of a data center is a fraction of the incoming capacity and that raises a blocking (bottleneck) problem. Performance of data centers relies heavily on the performance of networking infrastructure inside data centers. For instance, large-scale web applications such as searching are typically based on partitioning/aggregation scheme whose performance depends on the performance of the underlying network. Some applications are delay-sensitive and may require maximum bound on response time.

The main challenge is how to build a scalable DCN that delivers significant aggregate bandwidth with low latency. Many proposals such as fat-tree [4], BCube [5], DCell [6], PortLand [7], VL2 [8], Helois [9], and c-Through [10] were designed to meet the requirements of high performance DCNs in terms of throughput and delay with
lowest cost. The research community has begun searching on novel interconnected topologies to avail full bisection bandwidth which means providing full transmission rates between any two nodes in the network through commodity Ethernet switches. These proposals provide optimal switching capacity but require many links and switches that come with higher cost. Many of the recent topologies provide significant redundant paths among servers in the data center.

In this section, we first classify DCNs according to their connectivity and hierarchical structure. Then, we present a survey on the best current practices on DCNs. The main focus on modern DCNs is on their hierarchical structure and electronic switching design. Contrary, many recent proposals have explored using non-traditional network topologies that are not simply layered. They use switches and routers to build other network topologies such as DCell [6], BCube [5] and CamCube [11]. In these topologies, servers perform the routing functions.

2.2 DCNs Classification

In this section, we suggest a classification for data center networks. DCNs can be classified according to their connectivity and structure, where connectivity could be at the physical level or at the logical level (link-level). This classification covers 14 different DCNs topologies. At the physical level a DCN could be hierarchical, non-hierarchical, wireless, wired or hybrid. At the link level, a DCN topology can be build either on top of a hierarchical interconnection structure or an arbitrary structure. Figure 1 shows the classification of the most recent data centers followed by a brief description of each topology.
Hierarchical Structures

Hierarchical DCNs topologies adopt the Clos topology \[12\] to build a hierarchical structure for the network. The Clos topology can deliver high bandwidth using Ethernet commodity switches instead of using high-end switches. These topologies can efficiently lower the cost of building DCNs. However, they suffer from performance bottlenecks due to oversubscription at the higher layers of the hierarchy, which means allowing many incoming flows to share the same output port’s bandwidth resulting in higher latency. Using more expensive devices at the upper layers to provide more bandwidth might solve this problem.

Scaling is another drawback of this structure. They do not scale well for three reasons: First, servers are designed to be in a single layer-2 broadcast domain. Broadcast networks inherently suffer from scalability. Second, core and rack switches may raise
performance issues. Third, the number of ports on each switch limits scaling. Therefore, adding more servers while maintaining the physical structure would require replacing a large number of network devices as well as complex rewiring. In addition, hierarchical structures are highly vulnerable to a single point failure. For example, a core switch failure may crash thousands of servers. Switch oriented structures such as tree, Clos network [12] and fat-tree [4] cannot support many-to-many traffic properly.

2.3.1 Fat-tree

The Clos topology has a distinctive instance referred to as fat–tree or folded-Clos that interconnects Ethernet switches where the capacity of the network doubles as we go upward the tree as shown in Figure 2.

Fat-tree is a high bandwidth low latency topology. It is well-structured to support full bisection bandwidth without using expensive core switches. It requires moderate...
modifications to the forwarding functions of the switches. In this topology, there is no need to do any modifications to the end hosts. Fat-tree utilizes a central entity with full awareness of the cluster interconnection topology. Each switch in the network maintains a bidirectional forwarding detection session (BDF) with its neighbors so as to note any failure in the network.

Fat-trees have a specific IP addressing assignment. All IP addresses on the network get allocated within the private 10.0.0.0/8. A host’s address follows from the pod switch to which it is connected to; they have address of the form: 10.pod.switch.ID. where ID refers to the host’s position in the subnet. Routing tables need modification to allow two-level prefix look up, and they are implemented in content addressable memory to speed up the searching process. However, these routing tables increase look up latency.

Fat-tree employs a flow classifier to classify flows based on the IP addresses of the incoming packets for each source and destination. Thus, subsequent packets with the same source and destination depart from the same port to avoid packet reordering. Large flows are scheduled using a flow scheduler to prevent flows from overlapping with one another.

2.3.2 VL2

VL2 [8] is the network architecture that solves the issue of oversubscription. It creates an illusion of a large Level 2 (L2) network, above an IP network that implement any variation of the Clos topology. It scales to support massive data centers with a uniform high capacity between servers (i.e. high bisection bandwidth). Additionally, it provides the DCN with Ethernet layer 2 semantics. The notion of offering layer-2
semantics makes servers consider that they share a large single IP subnet. That eliminates the ARP and DHCP scaling problems that face the Ethernet. Besides, any services can be assigned to any server through flat addressing.

VL2 implements flow-level Valiant Load Balancing (VLB) technique [8], which spreads traffic randomly along paths. VL2 employs Equal-Cost Multipath (ECMP) forwarding in Clos topologies by assigning a single IP anycast address to all core switches.

VL2 topology faces a substantial number of challenges. It requires some modifications for hosting operating systems. Its deployment requires expensive switches. Unlike fat-tree topology, where every link is a cheap Gigabit Ethernet, this topology uses up to ten times fewer links in the upper layers as opposed to the others, but it uses 10 gigabit Ethernet for each.

### 2.3.3 Portland

The prime objective of Portland [7] is the delivery of scalable layer 2 routing, addressing and forwarding for DCNs. The switches may entirely plug and play in this topology without any administrator configurations to operate. Edge switches can automatically discover its unique pod number and physical location by exchanging a Location Discovery Protocol (LDP) messages. Therefore, the switches may entirely plug and play with the help of ToR switches.

The reason behind efficient routing and forwarding is through Virtual Machine (VM) migration in hierarchical pseudo MAC (PMAC) addresses. End hosts remain unmodified maintaining their actual MAC addresses (AMAC). Topology information of
a host is encoded into its PMAC with the form of pod:position:port:vmid. Hosts are responsible to perform ARP requests to get the PMAC of other destination hosts. That is, all packets forwarding takes place based on PMAC addresses, which require small forwarding tables in each switch. Egress switches perform PMAC to AMAC header rewriting to keep the illusion of unaltered MAC addresses at the destination hosts. Portland topology uses logical centralized fabric manager to maintain soft state about the network configuration details. It runs on a dedicated machine to assist with ARP resolution and fault tolerance. The disadvantage of this structure is that it is limited to the topologies that uses tree structure and does not support incremental deployment.

2.4 Random Structures

Due to the inherited limitations in the hierarchical topologies, researchers moved to other ways to interconnect DCNs more effectively. Random structures are primarily designed to avoid bottlenecks in the trees. These structures are inspired by other networks like small world and scale-free networks. The developers of these topologies have adopted solutions from these networks to overcome the problems in DCNs. From experiments [13] that have been conducted, the random flows spreading results in network utilization that is close to the optimal. This random structure does not limit all routes to a regular, rigid pattern like other structures. Many proposals fall in this class; namely, Jellyfish [14], Small World Data Centers [13], Scafida [15] and SPAIN [21].
2.4.1 Jellyfish

Scaling is one of the most important issues in today’s DCNs. This proposal focuses on incremental expansion problem. The proposed interconnection is a degree bound random graph topology on top of ToR switches. This design allows adding new network elements easily with more ports as needed. Unlike other structures, Jellyfish allows for construction of arbitrary network sizes with full bisection bandwidth and limited path lengths. It is also resilient to miss wiring and failures during construction. It employs a random topology adopted from random graphs as shown in Figure 3, for this reason, the graph is able to maintain its structure in case of failure.

Figure 3. Jellyfish topology with 16 servers.

In routing, the use of multipath TCP is coupled with a controlled routing scheme. This structure, however, does not apply to massive data center networks.
2.4.2 Small World Data Centers (SWDC)

In SWDC [13], hierarchical switches are avoided in favor of a richer topology in which data center nodes aid in routing. It proposes that network data center should be wired randomly to utilize these random paths as shortcuts. This topology is based on other structures that are easy to wire such as torus, ring, grid or cube and modify this regular structure with additional network links placed randomly throughout the data center. Figure 4 shows a SWDC that is based on ring topology.

![Small World Data Center with ring topology.](image)

SWDC finds the optimal shortest path for all destinations then route packets through inexpensive greedy geographical routing protocol. This method requires each server to maintain a record of its directly reachable servers. However because of the
greedy routing that does not maintain global routing information and picks local optimal routes, it may end up favoring a small number of random paths.

This topology adopts Kleinberg’s small world model [16]. This model assures that all nodes can be reached easily. SWDC provides reliable network scalability and it reduces path lengths between nodes. It supports content-based addressing for fast look-up. Comparing with hierarchical structures, SWDC can achieve higher bisection bandwidth and better fault tolerance. Experiments show [13] that SWDC can surpass CamCube on both bandwidth and latency. However, due to the long paths and switching delays, SWDC shows larger latencies than fat-tree [4] and Cisco [40] data centers.

2.4.3 Scafida

Scafida [15] is an asymmetric data center topology inspired by scale-free networks such as the Internet and World Wide Web. Scafida uses randomness to connect the nodes. They have small diameters (i.e., the shortest path between any two nodes) inherited by scale-free network, high fault tolerance and can also be scaled in smaller and less homogenous increments. The other DCN topologies like fat-tree[4], VL2 [8] and BCube [5] share a common property, which is the symmetry. That makes them hard to be extended in small quantities. However, due to lack of regularity in Scafida, it cannot efficiently support greedy and content based routing.
2.4.4 SPAIN

SPAIN [21] provides multipath forwarding through the use of (COTS) Ethernet switches on top of arbitrary topologies. It pre-computes a set of paths to utilize the redundant paths in a given network structure. Then it joins these paths to form trees, which are mapped to distinct VLANs. Unlike Portland [7] and VL2 [8], SPAIN can support arbitrary topologies, which allows SPAIN to be used without redesigning the whole physical network. It uses unmodified COTS switches with minimal end hosts’ modifications. It also supports incremental deployment, and requires no centralized controllers. In case of failure, SPAIN can detect the failure and modify end-to-end paths.

2.5 Hypercube (Hierarchical Cell) Structures

The networks in this class are server-centric. Examples include BCube [5] and DCell [6]. The hypercube structure reduces path length and improves bandwidth through rich inter-cell network connections. The interconnection networks in this class depend on numerous commodity switches connected using complex network patterns. They allow expansion to certain degrees, but require servers with free ports especially reserved for future expansions. The hypercube structure, however, requires complex cabling which is hard to maintain.
2.5.1 DCell

DCell [6] employs a recursively-defined structure to interconnect servers. It connects servers (in the same rack) to a local DCell that has much higher network capacity compared with the tree-based topologies. DCell does not create any bottleneck due to the distribution of the traffic across all links. This structure uses servers equipped with multiple network ports and smaller switches, to aid the construction of its recursively defined architecture. In Figure 5, a server is connected to several others through mini-switchs via communication links. These links are assumed to be bidirectional.

![DCell topology](image)

Figure 5. DCell topology.
DCell employs a decentralized routing protocol called DCell Fault-tolerant Routing protocol (DFR) that exploits the structure and handles a range of failures within its interconnections. DFR performs a distributed and fault-tolerant routing algorithm without using global states. Another single-path routing algorithm is implemented for unicast traffic by exploiting the recursive structure of the topology, which is called DCellRouting. Routing is performed by servers that are connected to other DCells. This server then routes the packets to another DCell server till the packets delivered locally within a DCell.

DCell has its own protocol suit, which provides functionalities similar to IP over the Internet. The protocol is implemented at an intermediate layer between IP and link layer, which is called Layer-2.5 DCN. In DCN, an IP address is used for end-host identification without participating in routing, and the current applications are supported without any modification. DCell was designed to have a fixed one-to-one mapping between IP and DCN addresses. The DCN protocol suite is implemented as a kernel-mode driver, which offers a virtual Ethernet interface to the IP layer and manages several underlying physical Ethernet interfaces.

A forwarding module was developed to receive packets from all the physical network ports. This module maintains a forwarding table that checks whether the next hop can be found upon receiving a packet. When the next hop is not found, DFR routing will be used to find the next hop, which is then cached in the forwarding table. When a link state changes, the forwarding table becomes invalid and recalculated by DFR.
2.5.2 BCube

BCube [5] was designed for shipping-container-sized modular data centers. This type of data centers gained acceptance due to its high degree of mobility, short deployment time and low costs. It was built based on DCell. The difference is that BCube deploys switches for faster processing with active probing to achieve load balancing. Figure 6 shows BCube with 4 pods.

![BCube Diagram](image)

**Figure 6. BCube with n=4.**

Like DCell [6], the servers have multiple network ports connected to multiple layers of commodity mini-switches. Besides, BCube uses more mini-switches and wires than DCell. The symmetric structure of BCube is a recursively-defined structure as well. There are multiple parallel short paths between any pair of servers to improve the fault tolerance, which is very important in modular data centers. This also will improve load balancing.

In BCube, servers are designed with multiple ports, whereas switches connect a constant number of servers. Servers in the same rack are connected to a switch, which allows servers communicate and transmit traffic for each other. Severs are also connected
to k other switches at the next level of the hierarchy. Unlike DCell [6] and CamCube [11], switches in BCube are only connected to servers and never directly connected to other switches and a hierarchical structure is used.

In BCube, packets are routed using a self-routing protocol, which is a path routing protocol that is designed to find the path from a source server to a destination server. BCube uses a routing protocol referred to as BSR (BCube Source Routing). It incorporates routing intelligence into the servers. By taking the merit of the multipath property with an active probing into the network, it balances the traffic and handles failures without link state distribution. The capacity in this topology decreases as the switch failure increases. BCube uses packet-forwarding engine, which decides the next destination of a packet by one table look up. The packet-forwarding engine comprises of two components: a neighbor status and the end packet forwarding procedures. The engine can be implemented in hardware in order to isolate the server system from packet forwarding.

2.5.3 3D Torus

The recent research has mainly focused on alternative data centers topologies aimed to achieve a high bisection bandwidth and improve the failure resiliency while providing low latency. In these systems, expensive network switches are removed, and the routing functions are taken to the data center servers, which are often equipped with additional Network Interface Controllers (NICs). This approach has its own limitation. They mainly rely on grid interconnections, which is not easy to set up.
2.5.4 CamCube

The CamCube project [11] identifies a design for a shipping-container-sized modular data centers; it replaces the switches-based topologies with a 3D torus topology inspired by the topology used for Content Addressable Network (CAN). In this topology, each server is connected directly to other six servers. All flows in CamCube traverse the network using direct connections; hence, no switches or routers are required. However, it suffers from large network hop counts.

A 3D key is used as a space coordinator where each server is assigned (x; y; z) which represents the server location within the 3D torus. It is inspired by use of the key based routing (i.e., a look-up method uses distributed hash tables to find the nearest server in terms of number of network hops) to merge the physical and the virtual topologies [11].

Unlike other data centers, which use single routing protocol, CamCube uses a base multi hop routing in conjunction with self-routing scheme. This feature helps services designers in implementing their own routing procedures to optimize applications’ performance. Servers are allowed to modify and intercept the packets as they rout them. Traditional TCP/IP is also used to find multiple short paths between source and destination. However, multiple paths are not disjoint and the various shared links create congestion and packet loss, which will decrease end-to-end throughput. In addressing this, CamCube uses a custom routing procedure that routes packets using link disjoint paths. It yields to around three disjoint paths all source-destination pairs. In case of a failure, the 3D key coordinates are remapped.
2.6 Wireless Structure

Even with the advances in DCNs design researches, wired networks involve high wiring costs, entail performance bottlenecks, and have low fault tolerance. Recent proposals such as Cayley [17] developed wireless technology to build DCNs. This new technology poses tradeoffs and challenges. Wireless networks are easy to setup and very flexible. They can be upgraded easily since no wiring is required. The most critical issues are about the structure and feasibility. This field requires more research and improvement in order to handle issues like interference, reliability and cost.

2.6.1 Cayley

Cayley [17] methodology involves building wire-free DCNs that is based on the 60 GHz RF technology. The interconnections are inspired by Cayley graphs [18] to provide dense interconnects. In this topology, all switching fabrics are avoided.

This topology has a special design where servers are stored in cylindrical racks organized in prism shaped containers. This design helps to divide data center volumes into two sections that are Intra-rack and inter-rack. The cylindrical racks utilize space efficiently and are generalized to a topology, which can be modeled as a mesh of Cayley graphs.

Cayley uses a geographical routing technique referred to as XYZ routing. Every host maintains only three tables, which determine the next destination of the packet. In traditional topologies, any failure can affect many servers in the DCN. Contrary, Cayley can overcome on any failure in the DCN by utilizing some of the alternative paths.
2.7 Hybrid

Many other proposals suggest deploying DCNs with a tree structure and oversubscribing links to minimize the cost. These oversubscribed links are vulnerable to the hotspot problem, which results from over-utilizing links. Traditional solutions proposed combining many links and switches with different multipath routing so that the core of the network can no longer be oversubscribed. New proposals suggest taking an advantage of wireless or optical networks and implement them in conjunction with the current topologies. For example, an optical circuit can hold very large bandwidth over the packet switching technology. This fact motivates the researchers to explore how optical circuit switching technology can improve DCNs. Of course, employing these techniques will increase the material cost and the interconnections complexity as well.

Other hybrid DCNs proposals have been introduced such as C-Through [20] and Helios [9]. Here we explain c-Through that was designed to alleviate data center networks problems such as oversubscription, hotspots and congested links. However, optical switching suffers from slower switching speed and high cost.

Some other proposals suggest using wireless links as a solution for oversubscription and hotspot problems. Thus, instead of combining many more links and switches with different multipath routing. For example, in Flyway topology [19], flyway links can significantly improve performance and increase the network capacity. However, there are some concerns that are not clear yet about using wireless in data center networks, like the cost of interference, signal leakage and reliability.
In general, optical and wireless interconnections can improve the performance of DCN by adding more capacity. However, they involved different tradeoffs. Wireless devices do not need wiring that simplifies DCNs upgrades, while optical fibers still need wiring.

2.7.1 C-Through (HyPaC)

C-Through (HyPaC) [20] a hybrid circuit and packet switching DCN architecture. This topology combines the traditional hierarchical packet switching with rack-to-rack optical circuit switching network, using fiber optic, and the optical switches to increase the bandwidth.

C-Through adopts a traditional hierarchical packet switching network using Ethernet switches in a tree structure. It uses an optical circuit switching network to provide connectivity among ToR switches in the trees. To de-multiplex the traffic onto packet or circuit switched network, a traffic control technique is required. Each server has a management daemon, which updates the kernel with the inter-rack connectivity. The traffic then de-multiplexed by the kernel to either optical or electrical paths. The traffic demand is estimated by increasing the per-connection socket buffer size and observing the end host buffer occupancy at the run time, which requires additional kernel memory for buffering. Such a design will reduce the cost of the optical components. Besides, these optical links can significantly add higher capacity to the DCN.
2.7.2 Flyways

Flyways [19] suggested a new hybrid DCN to alleviate the problem of overutilization resulting from bursty traffic, which can cause congestion and degrade a DCN throughput. This proposal incorporates the 60 GHz wireless technology that can provide extremely fast connectivity at low cost and add more capacity to the bandwidth. This combination can improve the performance of a DCN dramatically. In this topology, each Top-of-Rack (ToR) switch is equipped with one or more 60 GHz wireless devices, with electronically steerable directional antennas.

A central controller is employed to monitor the traffic patterns. The controller sets up flyways between ToR switches to provide more bandwidth as needed, and then makes appropriate routing changes to route traffic over flyways. Once the flyway links are removed, the traffic flows over the wired links. Thus, during flyway changes or flyway failures, packets are simply sent back over the wired network.

This topology employs a simple traffic demand estimation mechanism that uses a filter driver on servers to collect traffic statistics. These filters use a simple moving average of estimates from the recent past.

2.8 Comparison

A number of different topologies have been proposed in the literature [4-21] to meet the requirements for high performance data center networks. Each of which has its own advantages and limitations. The goal is to build DCNs with high throughput, low latency, scalability and reliability. Most of the recent DCNs are employing COTS Gigabit Ethernet switches to distribute the workload to a low cost hardware, rather than using
high cost networking elements. These cheap switches raised many problems in the network such as packet drops due to the small buffers in switches.

Fat-tree [4] has been used in the world leading large scale data centers such in Tokyo National Institute for Technology in Japan [42]. It gained this popularity because it has the lowest latency among all other DCNs as well as its nonblocking bandwidth characteristic [3]. Other DCNs such as CamCube might be more appropriate for storage systems due to the large number of hops as the network expands. Therefore, one can choose a DCN that can be utilized to meet the consumers’ needs. In general, DCNs topologies share a layered connectivity structure as described in Table 1. Figure 7 illustrates these layers.

**Table 1. DCNs layers.**

<table>
<thead>
<tr>
<th>Layer 3</th>
<th>Core IP Access/ Boarder Routers to span traffic across the network and serve as gateways.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 2.5</td>
<td>Optional illusion layer to provide agility (i.e., assign any service to any server)</td>
</tr>
<tr>
<td>Layer 2</td>
<td>Aggregation Layer: Gigabit Ethernet Aggregation switches to aggregate incoming and outgoing connections. Load balancer to balance the utilization across the network.</td>
</tr>
<tr>
<td>Layer 1</td>
<td>Access Layer: Gigabit Ethernet Edge switches attach the physical level to servers’ resources.</td>
</tr>
<tr>
<td>Layer 0</td>
<td>ToR switches connect all servers in the same rack</td>
</tr>
</tbody>
</table>
Figure 7. Illustration of DCNs layers.

DCNs structures can be switch-centric such as fat-tree, server-centric such as CamCube or combine servers and switches to do routing such as in BCube DCN [5]. Most DCNs routing schemes are customized to accommodate a specific structure. Each of these routing schemes has pros and cons. In switch-centric structures, switches cannot directly use the existing Ethernet switches without modifications and do not perform well under various traffic patterns (e.g., one-to-all, all-to-all). On the other hand, server-centric structures, like CamCube [11] rely on servers for routing and provide excellent
performance under all-to-all traffic patterns. For both structures, having too many paths can result in large look-up tables, which increases the latency. In addition, most of these links are not utilized except for the core layer. For example, in fat-tree, only 40% of links are utilized [2]. However, loss rates are higher at Aggregation layer due to the bursty traffic and self-similarity generated by servers.

In addition to the previous limitations in current DCNs, one of the major issues that is shared among all DCNs is employing TCP as a transport protocol in the network. TCP is widely used for most of the convention networks. However, several studies [25] [33] [43] show that TCP is not the best choice for high bandwidth networks. Due to its additive increase and multiplicative decrease policy, it takes too long to fill the pipe of high-speed networks even when multiple streams of TCP are used. This issue and its implications are discussed in details in Chapter 3.
CHAPTER 3

Transmission Control Protocol (TCP) Issues in Data Center Networks

3.1 Introduction

One of the design goals in DCNs is to create an economical DCN model that is easy to manage. That has resulted in developing data centers using Ethernet commodity switches to meet the design goals. However, these design goals put a limit on some application demands. Further, they cannot handle the traffic inside data centers. The current solutions propose modifying the current TCP to meet the requirements of the data centers.

The obvious choice for DCNs designers is the convention transmission protocol TCP. It is a three-decade well-understood protocol by systems specialists and developers. TCP has the ability to utilize the network as effectively as possible. For example, TCP will try to utilize any available bandwidth in the network and avoid congestion. Additionally, TCP guarantees fairness among all connections, support reliable transmission and congestion control. These features of TCP usually hold true in both WANs and LANs. Noteworthy, the characteristics of network traffic within DNCs are quite different from the characteristics of the traffic of other networks. In conventional networks, communications are spread over a large area whereas in data centers, nearly 80 percent of communications stay within the data center [3]. As a result, applying TCP without appropriate modification to data centers could degrade the network performance and waste network resources. To solve this issue, several researchers have proposed new
TCP protocols that are more appropriate for data centers and can support various traffic patterns.

Congestion is one of the problems that can affect the performance of DCNs. To reduce the possibility of congestion, the network is designed with redundant paths and high bandwidth to fit all traffic patterns. However, having too many networking components will increase the cost of the DCN.

TCP Incast is another common problem in data centers. Incast congestion happens when multiple sending servers under the same ToR switch send data to one receiver server at the same time [3]. This causes packet drop and retransmission for one or more TCP connections. Incast congestion causes TCP throughput to become severely degraded. One of the proposals [22] suggests enabling microsecond-granularity in TCP timeouts (RTO) to eliminate TCP incast collapse in data center networks. Another proposal suggests the design of incast congestion avoidance technique at the receiver side called Incast Congestion Control for TCP (ICTCP) [23]. This method adjusts TCP receiver window proactively before packet drops occur. This chapter investigates TCP issues followed by an overview of different TCP alternatives.

3.2 TCP Incast

TCP incast [27] congestion happens in high-bandwidth and low-latency networks when many synchronized servers connected to the same Gigabit Ethernet switch send
concurrent data to the same receiver in parallel as shown in Figure 8. This many-to-one traffic pattern is common in many important data center applications such as MapReduce and Search [25].

![Figure 8. TCP incast in DCNs.](image)

TCP incast phenomena may severely degrade DCN performance, e.g., by increasing response time due to packet loss and wasting network resources as well. The major cause of TCP incast phenomena is overflowing the Ethernet commodity switch buffer with a highly bursty traffic of multiple TCP connections in short intervals which results in severe packet loss and therefore TCP retransmission and timeout. These timeouts last hundreds of milliseconds on a network whose round-trip-time (RTT) is very small such as 100 or 200 microseconds [27]. Clearly, that wastes network resources, underutilize links’ capacity and could reduce throughput by up to 90% [22]. This phenomenon was first observed by Nagle et al. in [26].
The preconditions for TCP incast collapse [22] are as follows:

1. Data center network that is designed to achieve high-bandwidth, low-latency and utilizes commodity Ethernet ToR switches with relatively small buffers.
2. The existence of synchronized many-to-one traffic pattern.
3. The transmitted data is small for each request.

Recent data center networks are primarily well-structured and layered using (COTS) Ethernet switches. While these switches can provide full line-rate transmission, some essential features found in high-end switches are still missing. In specific, COTS typically have small buffers with small forwarding tables. Furthermore, they also normally employ simple queuing management techniques such as tail-drop. That is, using these switches can worsen the problem.

### 3.3 Congestion Trees

Hotspots can cause further complications in the network if not resolved. When a network becomes congested due to a hotspot, it will form a congestion tree with this hotspot as a root and all the paths from switches to the hotspot forms a congestion tree [28]. Congestion is more difficult to control when a hotspot forms a congestion tree.

Employing off-the-shelf Gigabit Ethernet networking components will increase network contention, which may lead to congestion that quickly spread across the network forming congestion trees. The absence of congestion control or congestion avoidance mechanisms may degrade the throughput severely when the network enters saturation.
3.4 TCP Outcast

Switches and routers with tail-drop queuing management scheme suffer from the port blackout problem. Port blackout occurs when two different large and small TCP flows arrive on multiple input ports of a switch and compete for the same output port. Subsequently, the large flows will be queued successfully in the output port while other smaller flows will be dropped. This phenomenon occurs when the switch uses a simple tail-drop queue management scheme [29]. The two main conditions for the port blackout problem to occur are as follows [29]:

1. Data center network that utilizes commodity Ethernet switches that employ tail-drop queue management scheme along with small buffers.

2. The existence of a large TCP flows and a small TCP flows arriving at two different input ports and compete for a bottleneck output port at a switch.

The major cause of TCP Outcast problem in data center networks is input port blackout at core switches happening due to the tail-drop discipline of the output queue. Consequently, throughput of the small flows that share the blackout output port will severely degrade. One of the proposed solutions [29] to solve TCP Outcast problem is to drop packets from all competing flows equally to avoid blackout of any flow.

3.5 Traffic Patterns within Data Center Networks

Understanding traffic patterns within DCNs is essential to optimize networks for data centers. In addition, understanding traffic characteristics of DCNs can lead to
advancement in traffic engineering schemes developed to utilize available capacity, reduce loss rates within DCNs, improve QoS and manage energy consumption.

DCNs traffic is regularly measured and characterized according to a sequence of packets from a source to a destination host, known as flows. Traffic from a client to a server is generally small whereas the response traffic from a server to a client tends to be larger. Of course this asymmetric traffic can varies according to the applications.

Traffic within DCNs has been classified as query (2-20 KB), short messages (50 KB – 1 MB) and large flows (1MB – 50 MB). Most of the flows in data centers are small (≤ 10KB) [3]. The most common classification of flows is using so-called mice and elephant classes. Elephant flows are long-lived flows with large number of packets. Those elephants result in a bursty traffic behavior, which can create a set of hotspot links that can lead to congestion trees and packet losses. This behavior can significantly affect the network performance. This phenomenon leads DCN designers to provision the network with a significant amount of oversubscription and bandwidth along with different load balancing techniques.

Data centers’ applications have a large impact on the traffic patterns in DCNs. Most of data centers applications are data and communication intensive. For instance, more than 1000 servers and analysis applications might interactively process petabytes of data in response to a simple web search request. According to the characteristics of data centers applications, researchers have defined three different traffic patterns: one-to-one, one-to-all, and all-to-all communications.
Communications within DCNs [3] can be from a DCN to a DCN, a DCN to user or within a DCN [1]. Internal traffic is the amount of traffic that stays within the DCN and external traffic, which are the traffic flows exiting core switches. In data centers, 40% - 90% of the traffic travel out of the data center [2]. Figure 9 shows a traffic classified by destination, adopted from CISCO Cloud Index 2011[1].

![Traffic classified by destination](image)

**Figure 9. Global data centers traffic by destination.**

A good data center traffic engineering solution should utilize the unique characteristics of data centers, instead of employing convention traffic engineering schemes that were originally designed for Internet and LANs. Data center networks have well-structured topologies which make them more symmetric with fixed locations for sources and destinations. In addition, compared with the Internet, there are usually more redundant paths between any two servers in DCNs.

Furthermore, because of the centralized management in DCNs, it is possible to monitor all the link status inside a DCN and manage the network accordingly. These
properties open the chances to find traffic engineering solutions for better utilization and to optimize DCNs performance as explained in the following section.

### 3.6 Proposed Solutions

Several proposals such as DCTCP [24], ICTCP [23] and MPTCP [30], suggest modifying TCP to be more appropriate for DCNs. These solutions are designed to provide better traffic control, congestion control and congestion avoidance. The following section discusses different proposals and how do they improve DCNs overall performance.

#### 3.6.1 DCTCP

Data Center TCP (DCTCP) [24] is a new transport protocol designed to meet the needs of applications in data centers. The goal of DCTCP is to achieve high burst tolerance, low latency, and high throughput using commodity switches. DCTCP addresses three major problems in TCP that are related to small buffer switches: incast, queue buildup and buffer pressure.

DCTCP operates with small queue without loss of throughput. It employs a simple marking technique at switches by marking Explicit Congestion Notification (ECN) field in the IP packets. In this scheme, the switch sets the Congestion Experienced (CE) bit of the packets as soon as the buffer occupancy of the switch exceeds a predefined threshold. The DCTCP source reacts by reducing congestion window by a factor that depends on the fraction of marked packets: the larger the fraction, the bigger the decrease factor is. That results in a smaller congestion window and reduced
transmission speed to alleviate queue congestion. This approach can effectively lower down the likelihood of overflow caused by incast. Regarding outcast problem, DCTCP can mitigate the effects of outcast problem since it ensures that queues do not become full.

### 3.6.2 ICTCP

ICTCP [23] is another approach designed to resolve incast problem in DCNs. The main idea is to adjust receiver window per connection on the fly according to the available bandwidth to avoid overwhelming the receiver. In this proposal, only receiver side of TCP is modified, hence, less burden to modifying TCP. ICTCP adjusts the receiver side of TCP window according to the ratio of the difference between the total of achieved and expected throughputs per connection over expected throughput and the last-hop available bandwidth. This approach can solve incast problem effectively [23].

The difference between ICTCP and DCTCP is that ICTCP only modifies the TCP receiver side, and ICTCP uses more complex approach to estimates the receiver window size and avoids congestion effectively. On the other hand, DCTCP seems to be a broader approach that provides efficient solutions for both incast and outcast.

### 3.6.3 MPTCP

Traditional DCNs adopt the idea of “one flow, one path” which cannot handle the traffic in data centers efficiently. This approach raises many problems such as collision, unfairness and limited utilization. MPTCP [30] proposed the idea of using multipath TCP for one flow to improve network performance by using parallel paths. In other word, it
creates more than one path between the same pair of endpoints for the same flow. MPTCP implements short timescale distributed load balancing. That provides better utilization of the available redundant paths in modern data center topologies and offers better performance without additional cost.

The protocol works as follows. First, TCP connection starts to negotiate MPTCP support in the initial SYN exchange. Then, additional sub-flows can then be opened. Once multiple sub-flows have been created, the sending host’s TCP divides data between the sub-flows. Other TCP options allow the receiver to reorder the received data. One of the most important features is that MPTCP does not require applications to be aware that MPTCP is being used instead of TCP. MPTCP seems to mitigate unfairness among different TCP flows successfully.
CHAPTER 4

Routing Schemes

4.1 Routing in DCNs

Many routing protocols have been developed for enterprise networks such as OSPF/ISIS (link-state routing) or BGP (path-vector routing). Using these protocols for DCNs may result in suboptimal performance for several reasons. First, DCNs have many different characteristics than other convention networks. Second, routing within well-structured topologies can be usually done within fewer hops and shorter distances. Third, DCNs have significantly higher link density. Forth, the communications within data centers cannot widely spread like the communications in other networks. Lastly, traffic patterns in data centers can highly affect their performance. Therefore, there should be routing techniques to reduce the complexity of calculating the available routes within a DCN.

Many routing optimization techniques have been involved to optimally utilize DCNs resources, achieve full bisection bandwidth and avoid congestion. In order to fully utilize the network resources and capacities within the physical structure, researchers have proposed several routing schemes. For example, most of the current DCNs such as fat-tree [4], BCube [5] and DCell [6] employ a customized single routing protocol to route packets within the topology. Another approaches suggested using double routing scheme such as CamCube [11]. CamCube implements a link-state routing protocol that
exploits the multipath characteristics and routes packets using shortest paths while servers are allowed to interrupt and modify packets as needed. This flexibility gives the services designers the ability to optimize the performance requirements for different services using their own customized routing algorithms.

4.2 Routing Optimization

There are several proposals where more advanced routing schemes have been developed. These routing schemes can be either simple such as Equal-Cost Multi-Path (ECMP) [32] or complex such as Hedera. ECMP employs a simple path assignment technique that uses multiple paths for each connection. Valiant Load Balancing (VLB) [8] is another routing scheme that distribute traffic load across network paths randomly. Both ECMP and VLB assign paths blindly regardless of workloads and utilization which can lead elephant flows to collie. Another sophisticated approach is Hedera [31], which employs a central scheduler with a global knowledge of each link-state to distribute the large flows dynamically across different paths. Another important issue in DCNs routing is load balancing to achieve optimal resource utilization, maximize the throughput and minimize the response time while avoiding congestion. Most DCNs use the state-of-the-art hash-based Equal Cost Multipath routing protocol (ECMP). Some other proposals have shown that a centralized management approach can outperform ECMP. Routing Approaches in DCNs

While most basic routing schemes aims to find shortest paths with low latency and high throughput. Routing in DCNs requires further optimization to utilize network resources efficiently with the least latency and highest throughput as well as reliability.
We classify routing schemes among different DCN structures as centralized and distributed routing.

4.3 Centralized routing

4.3.1 Hedera

In some approaches, adaptive routing has been implemented with a centralized traffic controller such as Hedera [31] where edge switches detect and report large flows to the centralized controller. Hedera complements ECMP to deal with communication patterns that cause problems for ECMP. When a large flow is detected, the centralized controller explores a new path and updates switches’ routing tables dynamically. The centralized controller has a global view of links status and traffic demands so it can see all the bottlenecks. Hedera is one of the most efficient techniques that benefit hierarchical DCNs if implemented.

A centralized traffic controller limits the scalability. Additionally, it can be a potential point of failure. In other words, when a datacenter incrementally expand, the traffic control messages sent to and from a centralized controller may congest the links that connect the controller to the data center. Hedera can successfully fully utilize bisection bandwidth of a network, but a recent research suggests that this centralized approach needs parallelism to speed up routes computations [43].
4.4 Distributed Routing

Other DCNs adopt distributed routing approaches, which are more scalable. However, they may create hot spots due to the absence of workload on each path. Below is a description of different distributed routing techniques.

4.4.1 Equal-Cost-Multi-Path (ECMP)

ECMP is a distributed routing technique that works at flow-level performed by switches. ECMP-enabled switches are configured with multiple paths for a given destination. It can split traffic to each destination across multiple paths according to a hash value for each path without the need for packet reordering at the destination. The drawback of ECMP is the elephant flows as they can collide on their hash values and result in congested output port.

4.4.2 Valiant Load Balancing (VLB)

VL2 [8] is a topological structure of DCNs. This topology employs Valiant Load Balancing technique (VLB), which is another distributed path selection mechanism. It aims to achieve full bisection bandwidth under any traffic pattern while utilizing available bandwidth. VLB places a path selection algorithm at edge switches. Edge switches forward the flows by selecting a random core switch which then forwards the flows to the last destination.

VL2 has major drawbacks in its implementation; the randomness fails to manage network resources efficiently because it does not guarantee that elephant flows will not collide. Additionally, it cannot achieve full bisectional bandwidth because some flows
will randomly choose the same path while other links randomly fail to be selected. VLB can be only used for DCNs with hierarchical structure such as fat-tree.

4.4.3 DARD

DARD [33] adopts distributed dynamic path selection approach. It differs from ECMP and VL2 in two major aspects. First, it selects the path dynamically according to the traffic load. Therefore, DARD can detect elephant flows and distribute them appropriately. Second, it can be implemented in server-centric structures in which flows are routed by servers. Each server monitors path states to distribute the traffic equally across available paths. DARD is easy to deploy since it does not require switches upgrade, upgrading servers’ software is enough to optimize network performance using DARD.

4.5 Load-Balancing Techniques

The trend of DCN topologies is to use much denser interconnects to achieve high bisection bandwidth for any traffic patterns to fulfill applications demands. However, while this dense interconnection can resolve some issues in DCNs, it raises difficult challenges for routing such as larger routing tables that slow down look-up speed. Achieving a full bisection bandwidth requires the load to be distributed between available paths evenly.

To ensure the high performance and guarantee a full bisection bandwidth, load-balancing techniques are essential. Load balancing mechanisms are employed to spread
traffic across the existing links in the network. Some DCN topologies implement its own load-balancing scheme such as VL2 and DCell.
CHAPTER 6

Case Study: Fat-tree Topology

One of the widely used network topology is fat-tree [4] which is basically a hierarchical structure where the capacity of the network increases for traffic aggregates upward. Figure 10 illustrates the topology of a binary fat-tree network. This topology is used and implemented in largest data centers worldwide, mainly due to its ability to provide full bisection bandwidth as well as very low latency. However, hierarchical structures in general have the saturation problem as we go upward toward the root, which limits the overall performance of the network.

6.1 Theoretical Background on Fat-tree

Fat-tree is a variation of the Clos network [12], which has been extensively studied as a Multistage Interconnection Network (MIN). It was originally designed to meet the performance requirements of on-chip interconnection networks. It provides near optimal performance in that sense. Multistage interconnection networks for multiprocessors are the most complex component of the system. Theoretically, the ports of the switches limit incremental expansion of such a system.

Fat-tree was first proposed by Charles Leiserson in 1985 [34] to interconnect the processors of a supercomputer. It has been shown [34] that given a network size, fat-tree gives the optimal routing network for that size. For any given volume of hardware, no network outperforms fat-tree [34].
Recently, almost all large-scale commodity high performance computing clusters are interconnected using fat-tree topology due to the fact that it can effectively utilize any given amount of hardware resources devoted for communications.

Fat-tree is a folded non-blocking rearrangeable Clos [12] network with a centralized controller. That is, a new connection can always be established from any idle input port to any idle output port by rearranging the existing connections [13]. The topology can support any permutations of the communications without network contention. Therefore, it can achieve full bisection bandwidth for any input/output permutation and performs as a crossbar switch.

6.2 Approximated Fat-tree

Fat-tree topologies that are used in DCNs are approximation of the original fat-tree network. The original fat-tree might not be appropriate for a communication network environment due to the large nodal degree of the root [35]. Alternative topologies have
been proposed to solve this issue by adding more nodes with smaller nodal degrees instead of having one root with a large nodal degree. Figure 11 shows the approximated fat-tree network. This approximation trades off connectivity with simplicity.

![Approximated fat-tree topology](image)

**Figure 11.** Approximated fat-tree topology.

### 6.3 Original Fat-tree and Approximated Fat-tree in Communication Network Environment

The original fat-tree was designed to perform optimally under three conditions: a uniform traffic, centralized management and a tight nonblocking condition. Contrarily, conventional nonblocking networks become blocking in the computer communication environment due to the following three important factors.

1. Nonuniform traffic in data centers

The original fat-tree was designed to perform optimally under a uniform traffic. However, the traffic nature in data center networks has its own characteristics that are
different than the traffic characteristics in other networks. When the demand for different services is nonuniform or exceeds the network’s capacity, a congestion tree will build up and severely degrades the network throughput. Dealing with these issues makes the traffic engineering more complicated in a communication network environment.

2. The absence of a scalable centralized routing mechanisms

The centralized controller might limit the scalability of a DCN. As a result, most of the proposed DCNs implement a distributed routing scheme.

3. Nonblocking condition

It is not effective to build nonblocking fat-tree with small top-level switches. Therefore, to determine the right implementation of fat-tree that guarantees optimal performance with the lowest cost while supporting any communication permutations with full bisection bandwidth, there should be a predefined nonblocking condition. For fat-tree with single-path deterministic routing, the nonblocking condition is defined in [36] as follows.

**Theorem:** A fat-tree consists of m-ports n-trees with r × r internal switches, where \( r \geq 2n + 1 \), denoted as FT(m, n, r ), is non-blocking if and only if \( m \geq n^2 \).

Building a fat-tree network with appropriate values for n, m, and r results in a cost-effective network that guarantees nonblocking communications for any sets of input-output permutations. Since nonuniform nature of the traffic in communication networks requires some degree of adaptivity to smooth out the traffic and provide higher
throughput, it is important to consider determining the tight nonblocking condition for fat-tree with adaptive routing, which is still an open problem.

### 6.4 Bottlenecks in Fat-tree

There is an inherent bottleneck in fat-tree topology, which is related to the shape of core level switches. In fat-tree, there are two types of switches: $n \times n$ switches and $n \times m$ switches where $m < n$. Core switches are $n \times m$, whereas other aggregation and edge switches are either $n \times n$ or $n \times m$ where $m > n$ as illustrated in Figure 12.

![Figure 12. Concentration in fat-tree topology.](image-url)

At Aggregation layer (Fat section) and Edge layer (Slim section) as shown in Figure 13, each input is connected to at most one output. Therefore, no two outputs share an input and the connection pattern allows only one input to access one output, hence, all $n$ packets are winners. On the other hand, switches at the core layer (Ladder section) have the shape of packet concentrator switches. Assuming that $n$ is the number of input lines in the switch with $v$ arrival rate and $m$ is the number of output ports with $w$ departure rate,
therefore, if \( vn > mw \) then \( vn - mw \) packets are going to be dropped. As a result, those concentrator switches at the core layer are considered as a bottleneck in fat-tree topology. Those concentrator switches form a contraction point in the topology as shown previously in Figure 12. That causes the performance to degrade, particularly when they are highly utilized and become prone to hotspots. Consequently, that results in higher latency in DCNs that negatively affect all applications and quality of service as well.

![Fat-Tree Abstraction](image)

**Figure 13. Abstraction of fat-tree.**

Expanding the switches to have more bandwidth will not solve the problem because this is inherent in all hierarchical structures. Optimizing the routing techniques might be more efficient to distribute the traffic based on the network workload.

### 6.5 Utilization in Fat-tree

There is a growing demand in research community to develop new routing techniques that are able to support various applications within the same DCN optimally.
Understanding traffic characteristics and the distribution of workload helps researchers to make better choices regarding routing techniques such as determining what strategies should be used for load balancing and how to deploy different services.

Recent studies [3] have shown that network utilization is significantly high at the core layer, whereas 40% of links in the aggregation layer are never used. This is expected since the bandwidth at the aggregation layer is 4 times more than the bandwidth at the core layer. This significant underutilization at the aggregation layer exists in most of DCNs topologies with hierarchical structure such as fat-tree. Links at the edge layer have higher utilization than links at aggregation layer because of their lower bandwidth. Given the fact that there is a large number of links in fat-tree, an optimal traffic engineering scheme can distribute traffic dynamically across the links based on links utilization.

6.6 Theoretical Analysis of End-to-End Delay in Fat-tree Topology

Fat-tree can be modeled by forming an open tandem queuing model in which each node has a FIFO buffer of infinite size and packets pass through a series of rely nodes towards a sink node. In an open queuing network, packets enter and depart from the network. In tandem queuing systems, packets arrival times to the receiving node are correlated with packets departure time from the preceding nodes. Such networks of queues are used to model and analyze potential contention and tandem queuing performance. According to Kleinrock independence approximation, which states that integrating several packet streams on a single transmission line, has an effect similar to restoring the independence of interarrival times and packet lengths. Therefore, an M/M/1 model is appropriate to analyze the behavior of each communication link.
Figure 14. A node configuration with aggregate traffic.

Figure 14 shows a partial structure of the network in Figure 11. Both external and relay flows are aggregated with cumulative rate entering node j. Suppose $r_{ij}$ is the probability of a packet routed from node i to node j. A packet departing from node i may arrive to node j deterministically with probability $r_{ij} = 1$.

Let $R$ be the $n \times n$ probability matrix describing the routing of packets within a Jackson network [44]. Let $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$ be the mean arrival rates of the relayed packets, and $\gamma = (\gamma_1, \gamma_2, ..., \gamma_n)$ be the mean arrival rates of the external packets.

The rows of $R$ matrix need not necessarily sum up to one, such that $\sum_j r_{ij} = 1$. A Jackson queuing network is a network of an $n$ M/M/n state-independent queuing system with the following features. (i) There is only one class of packets arriving to the system. (ii) External packets arrive at node j according to a Poisson process with rate $\gamma_j \geq 0$. (iii) The service times of the packets at $j^{th}$ queue are exponentially distributed with mean $1/\mu_j$. Upon receiving its service at node i, the packet will proceed to node j with a probability $r_{ij}$ or leave the network at node i with probability $(1 - \sum_{j=1}^{n} r_{ij})$. Finally, the queue capacity at each node is infinite, so there is no blocking. Assuming the network reaches a steady state, we can then write the following traffic equation using the flow conservation
principle, in which the total sum of arrival rates entering the system is equal to the total
departure rate under steady-state condition.

\[ \lambda_j = \gamma_j + \sum_{i}^{n} \lambda_i n_{ij}, \quad j = 1, 2, \ldots, n. \] (1)

Assuming the network is stable, the aggregate input rate \( \lambda_i \) into node \( j \) is equal to
the aggregate output rate from node \( i, i = 1, \ldots, n \). Therefore, we have a system of \( n \)
equations and \( n \) unknowns. These equations can be written in matrix form as,

\[ \vec{\lambda} = \vec{\gamma} + \vec{\lambda} R \] (2)

and the aggregate arrival rate vector is solved by,

\[ \vec{\lambda} = \vec{\gamma} (I - R)^{-1} \] (3)

where, \( \gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n) \) and the components of the vector \( I \) give the arrival rates into
the various stations. It is shown in [44] that \( (I - R) \) is indeed invertible. Assuming the
external arriving traffic are independent and identically distributed (iid) Poisson process
with rate \( \gamma_j \), and the service time at node \( j \) is assumed to be exponentially distributed with
rate \( \mu_j \), \( j = 1, 2, \ldots, n \). The service times are assumed to be mutually independent and
also independent of the arrival process at that queue, regardless of the previous service
times of the same packet in other nodes. After the net rate into each node is known, the
network can be decomposed and each node can be treated as an independent queuing
system with Poisson input. This also follows that the average number of packets and the
average delay in each queue is the same as the corresponding M/M/1 queue. To ensure the
steady state distribution of the network model, it is necessary that the aggregate packet
arrival rate \( \lambda_j \), computed in Equation (1), be less than the service rate \( \mu_i \), that is \( \lambda_j < \mu_j, j = 1, 2, \ldots, n \), such that
\[ \tilde{y} (I - R)^{-1} < \tilde{\mu} \] (4)

This is all needed to check the stability of the network. Assuming \( \lambda_j < \mu_j \) for \( j = 1, 2, \ldots, n \), define \( \rho_j = \lambda_j / \mu_j \). \( \rho_j \) identifies the fraction of the time the transmitting buffer of node \( j \) is not empty, i.e., node \( j \) has a packet to transmit. Jackson’s Theorem [44] also states that the joint steady state distribution for the number of packets in each node is given by:

\[
\Pr[L_1 = n_1, L_2 = n_2, \ldots, L_n = n_n] = \Pr[L_1 = n_1] \times \Pr[L_2 = n_2] \times \ldots \times \Pr[L_n = n_n] = \prod_{i=1}^{n} (1 - \rho_i) \rho_i^{n_i}
\]

where, \( L_i \) is the length of the queue at node \( i \) and \( n_i \) is a random integer.

### 6.6.1 End-to-End Delay Analysis for Downlink Traffic (Client-Server)

Analysis of end-to-end delay performance of the network is an essential measure for the quality of service (QoS). There are four delay components that contribute to the end-to-end delay of a connection; namely: access delay, queuing delay, propagation delay and transmission delay. Access delay and queuing delay can be measured probabilistically while propagation delay (distance) and transmission delay (bit-rate) are deterministic. While theory supports any size of network, in this section, we use a 4 \( \times \) 16 fat-tree to calculate the two probabilistic delay components for downlink traffic. The same analysis is performed in the next section for uplink traffic.
Figure 15. Downlink traffic (client-server) in fat-tree.

Figure 15 illustrates the direction of the incoming traffic as it goes from clients to servers. Traffic in this direction does not face any contraction point since the number of output-ports at each switch are either equal to input-ports or larger. For that reason, fat-tree has a very low latency with downlink traffic. Figure 16 shows the equations that are used to calculate average arrival rates ($\lambda$) for downlink traffic at each node. The probabilistic end-to-end delay at each node and for an end-to-end connection can be calculated using equations in Table 2.
<table>
<thead>
<tr>
<th>$\rho$</th>
<th>End-to-End Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>$\frac{97513}{30891168} \delta$</td>
</tr>
<tr>
<td>0.2</td>
<td>$\frac{20671}{5225376} \delta$</td>
</tr>
<tr>
<td>0.3</td>
<td>$\frac{7921}{1241856} \delta$</td>
</tr>
<tr>
<td>0.4</td>
<td>$\frac{15013}{1873248} \delta$</td>
</tr>
<tr>
<td>0.5</td>
<td>$\frac{8263}{766528} \delta$</td>
</tr>
<tr>
<td>0.6</td>
<td>$\frac{2819}{216384} \delta$</td>
</tr>
<tr>
<td>0.7</td>
<td>$\frac{4303}{288960} \delta$</td>
</tr>
<tr>
<td>0.8</td>
<td>$\frac{1171}{71136} \delta$</td>
</tr>
<tr>
<td>0.9</td>
<td>$\frac{413}{21120} \delta$</td>
</tr>
</tbody>
</table>
Figure 16. Aggregate arrival flows for downlink traffic.
### Table 2. End-to-end delay for downlink traffic

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>End-to-End Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>$\frac{97513}{30891168} \delta$</td>
</tr>
<tr>
<td>0.2</td>
<td>$\frac{20671}{5225376} \delta$</td>
</tr>
<tr>
<td>0.3</td>
<td>$\frac{7921}{1241856} \delta$</td>
</tr>
<tr>
<td>0.4</td>
<td>$\frac{15013}{1873248} \delta$</td>
</tr>
<tr>
<td>0.5</td>
<td>$\frac{8263}{766528} \delta$</td>
</tr>
<tr>
<td>0.6</td>
<td>$\frac{2819}{216384} \delta$</td>
</tr>
<tr>
<td>0.7</td>
<td>$\frac{4303}{288960} \delta$</td>
</tr>
<tr>
<td>0.8</td>
<td>$\frac{1171}{71136} \delta$</td>
</tr>
<tr>
<td>0.9</td>
<td>$\frac{413}{21120} \delta$</td>
</tr>
</tbody>
</table>

#### 6.6.2 End-to-End Delay Analysis for Uplink Traffic (Server-Client)

Another important traffic to consider is the traffic travelling from servers to clients in uplink direction. In this model, traffic will go through a concentration stage at the core level which results in increased end-to-end delay. As in the previous section, we
first calculated the expected aggregated arrival rate at each node as shown in Figure 17.

Figure 17. Uplink traffic (server-client) in fat-tree.

Equations in Figure 18 can be used to calculate the aggregate arrival rate for uplink traffic while equations in Table 3 can be used to find the probabilistic end-to-end delay. We calculated end-to-end delay for fat-tree uplink and downlink traffic under various traffic loads as shown in Figure 19. From these results, we conclude that downlink traffic has a very low delay while external traffic suffers more delay.
Figure 18. Aggregate arrival flows for uplink traffic.
Table 3. End-to-end delay for uplink traffic.

<table>
<thead>
<tr>
<th>ρ</th>
<th>End-to-End Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>( \frac{184927}{14281344} \delta )</td>
</tr>
<tr>
<td>0.2</td>
<td>( \frac{91807}{5619744} \delta )</td>
</tr>
<tr>
<td>0.3</td>
<td>( \frac{2911}{108192} \delta )</td>
</tr>
<tr>
<td>0.4</td>
<td>( \frac{9767}{284544} \delta )</td>
</tr>
<tr>
<td>0.5</td>
<td>( \frac{11021}{232232} \delta )</td>
</tr>
<tr>
<td>0.6</td>
<td>( \frac{5147}{87584} \delta )</td>
</tr>
<tr>
<td>0.7</td>
<td>( \frac{173}{2520} \delta )</td>
</tr>
<tr>
<td>0.8</td>
<td>( \frac{847}{10944} \delta )</td>
</tr>
<tr>
<td>0.9</td>
<td>( \frac{3593}{37440} \delta )</td>
</tr>
</tbody>
</table>
Figure 19. End-to-end delay comparison for fat-tree.
CHAPTER 7

Simulation

7.1 Models

In this thesis, we have simulated two fat-tree data center networks that are different interconnections density and compared its performance in terms of delay and throughput. We have simulated fat-tree data center network and Cisco data center network. Noteworthy, Cisco data centers are built on top of a fat-tree topology. However, the Cisco networks have less interconnection density than fat-tree. We simulated these networks using OPNET [41], which is a discrete event simulator.

The accuracy of the simulation was our first concern. Therefore, we have spent a considerable amount of time attempting to produce a valid and accurate simulation for the data center networks we are investigating. We evaluated several simulators to find out the most appropriate simulator for our study.

7.2 OPNET the Simulator

OPNET [41] is widely used in research and development. It comes with a large number of tools and predefined devices and utilities. This feature makes it very flexible. Despite the graphical user interface, working on OPNET requires a steep learning curve. However, it provides concrete and reliable simulations along with variety of devices and
models that can be dragged from an object palette in OPNET to the work area. Therefore, it was a suitable simulator to model and simulate DCNs. OPNET simulation cycle is illustrated in Figure 20.

![OPNET Modeler simulation cycle](image)

**Figure 20. OPNET Modeler simulation cycle.**

We used OPNET Modeler to implement the considered DCNs architectures. The most important features of OPNET Modeler are as follows:

i. Its university/academic relation and support.

ii. Comes with a rich set of switching commercial modules.

iii. Has a user-friendly interface.

iv. An implementation of real IP address.

v. The ability to build DCN using the same switches used in the real world.

vi. Real network bytes are contained in simulated packets

vii. Various scenarios can be tested reliably with good performance visualization tools.
7.3 Data Center Networks Models

Two data centers core architectural models, namely fat-tree topology [4] and Cisco DCN topology [40], have been used to analyze hierarchical DCNs performance. These models have been adopted to analyze the performance of different interconnections and application classification approaches.

7.3.1 Fat-tree DCN Model

To simulate traffic, we built the fat-tree model which consists of three switching layers that have been interconnected as illustrated in Figure 21. At the top of the model, four sink nodes were connected to the root of the tree and other 16 leave nodes to represent traffic sources. All switches for both models are Gigabit Ethernet switches. For Edge layer, all connections for both models are full duplex links with 1 Gigabits per second transmission rate and 10 Gigabits for all other links. Table 4 shows general settings for the fat-tree simulation.

To simulate applications, four servers were used to host four different applications; namely one Web Server, two Database servers, one FTP server and one Video conferencing server. Parameters that were used for these applications are described in Table 6. For the client side, a LAN network with 10 clients was used as shown in Figure 23. Regarding Network layer, realistic IPv4 addresses were automatically assigned for each network interface in the model.
Table 4. Fat-tree simulation parameters.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of pods</td>
<td>4</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>20</td>
</tr>
<tr>
<td>Time of simulation</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Packet size</td>
<td>1500 KB</td>
</tr>
</tbody>
</table>

7.3.2 Cisco DCN Model

To simulate traffic within Cisco DCNs, Cisco DCN model was configured with four switching levels with Gigabit Ethernet switches as shown in Figure 22. The sink nodes were represented using two servers and four other servers were traffic sources. All connections are full duplex links with 1 Gigabits/s for links at Edge layer and 10 Gigabits/s for all other links. As fat-tree model configuration for application simulation, the same applications and settings were configured for Cisco model as shown in Figure 24. The parameters that were used in the simulation of Cisco are given in Table 5.

Table 5. Cisco DCN simulation parameters.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of pods</td>
<td>2</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>6</td>
</tr>
<tr>
<td>Time of simulation</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Packet size</td>
<td>1500 KB</td>
</tr>
</tbody>
</table>

Other Simulation Parameters:

1. Routing

   For both architectures, OSPF routing protocol has been used.

2. Traffic pattern
For both topologies, one-to-many and many-to-one traffic patterns have been implemented.

3. Seeds

OPNET uses a generated random number to create a more realistic simulation. A seed value of 120 was used for all of these tests.

4. Duration

To look at the system in a steady-state performance, the results of the initial part of the simulation are not included. The transient state was removed by using very long runs besides multiple runs.

The performance of the considered models has been verified by using the theoretical analysis model.

7.4 Traffic Modeling

The main objective of this study is to identify bottlenecks with the proposed models as well as trade-off analysis between network performance and redundancy in network equipment. From theoretical point of view, the core layer as well as bursty traffic are the main sources of the network bottlenecks. Therefore, analyzing network performance is essential to determine the impact of these bottlenecks. We simulated M/M/1 traffic for two reasons. First, it allows us to compare the impact of M/M/1 traffic distributions on both simulation and theoretical models. Second, given the exponential back-off algorithm in the Ethernet, it is reasonable to use M/M1 to reflect the exponential nature of the service time. To experiment with various traffic loads (ρ), we varied the
packet inter-arrival time ($\lambda$), such that $\rho = \frac{\lambda}{\mu}$, where $\mu$ is chosen as a constant (1500*10,000 bytes/s) to reflect the transmission time of a packet at a switching port.

Figure 21. Fat-tree modeling using OPNET.

Figure 22. Cisco modeling using OPNET.
7.5 Applications Simulation

We simulated applications with heavy load and light load for both fat-tree and Cisco DCN to find out how interconnection density can improve the network performance. The parameters for applications are shown in Table 6. Both fat-tree and Cisco DCN were examined under those workload. Fat-tree modeling is shown in Figure 23 and Cisco DCN modeling is shown in Figure 24.

Table 6. Workload settings for applications simulation.

<table>
<thead>
<tr>
<th>Application</th>
<th>Heavy Load Parameters</th>
<th>Light Load Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web-browsing</strong></td>
<td>HTTP 1.1</td>
<td>HTTP 1.1</td>
</tr>
<tr>
<td></td>
<td>Page interarrival time: exp(60)</td>
<td>Page interarrival time: exp(720)</td>
</tr>
<tr>
<td></td>
<td>Page size (bytes): constant (1000)</td>
<td>Page size (bytes): constant (500)</td>
</tr>
<tr>
<td></td>
<td>Pages per server: exp (10)</td>
<td>Pages per server: exp (10)</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Transaction Interarrival time: exp(12)</td>
<td>Transaction Interarrival time: exp (30)</td>
</tr>
<tr>
<td></td>
<td>Transaction size (bytes): constant (32768)</td>
<td>Transaction size (bytes): constant (16)</td>
</tr>
<tr>
<td><strong>Application Server: FTP</strong></td>
<td>Inter-request time: exp (360)</td>
<td>Inter-request time: exp (3600)</td>
</tr>
<tr>
<td></td>
<td>File size (bytes): constant (50000)</td>
<td>File size (bytes): constant (1000)</td>
</tr>
</tbody>
</table>
Figure 23. Applications in fat-tree DCN.

Figure 24. Applications in Cisco DCN.
7.6 M/M/1 Traffic Simulation Results

The observations of M/M/1 queue models performance for fat-tree and Cisco DCN under different utilization rates ranging from 0.1 up to 0.9 show a comparison of the average network delay (ms.) for both DCNs. As shown in Figure 25, fat-tree average delay is lower than Cisco DCN average delay. Average throughput for both models was obtained under different utilization rates as shown in Figure 26. Fat-tree throughput was higher under higher utilizations.

![figure 25](attachment:figure25.png)

**Figure 25.** Average delay for fat-tree and Cisco DCN.

![figure 26](attachment:figure26.png)

**Figure 26.** Average throughput for fat-tree and Cisco DCN.
7.7 Applications Simulation Results

Under the same workload, fat-tree shows higher latency. However, fat-tree throughput was higher than Cisco DCN as shown in Figure 27.

![Figure 27. Applications' delay for fat-tree and Cisco DCN.](image)

Regarding throughput, fat-tree has a slightly higher throughput than Cisco DCN as shown in Figure 28.

![Figure 28. Applications' throughput for fat-tree and Cisco DCN.](image)
in Figure 28.

The simulation shows that fat-tree outperform Cisco DCN in terms of throughput and latency. Cisco DCN interconnection has more switching layer while fat-tree has more links density. From the simulations results, we can conclude that having more links can improve the network performance better than having more layers or stages.

7.8 Findings

Fat-tree and Cisco data centers have the same basic topology which is fat-tree or folded-Clos network. The main difference is that Cisco DCN uses fewer high-end switches and links whereas fat-tree data center has denser interconnections with more commodity switches. The extra switches and links in fat-tree provide better performance in terms of throughput and delay. However, that also can impose significant wiring complexity in large-scale networks as well as higher cost.

From Cisco DCN point of view, using fewer switches and links to interconnect a data center can efficiently utilize the network resources with less wiring complexity and lower down the cost. However, this approach might cause higher oversubscription ratios, more hotspots in the network due to the high utilization, which make it more prone to congestion and packet drop.

Our simulations results show that there is a trade-off between cost and performance, and that supports our findings. The relationship between cost and performance is more complicated. Since building a fat-tree with a few equipment can utilize network resources efficiently, building the same fat-tree with more redundant paths and more switches can result in better performance with underutilization of network
resources. Therefore, defining a nonblocking condition for fat-tree is essential to
determine the optimal deployment of this topology with the lowest cost
CHAPTER 8

Conclusion

8.1 Conclusion

This thesis provides a survey of most recent DCNs technologies with their classification based on their topologies and an overview of different structures and how are they deployed. We investigated the performance of TCP, in terms of delay, in data centers.

Even though many DCNs topologies have been proposed, neither one solves the current issues of data centers. Data centers networks became the bottleneck of the computation speed. From this prospective, this thesis built the basis for a comparative analysis and the issues that might affect data center networks including the two most important performance metrics; delay and throughput.

We also provide a theoretical background on the most important topology, fat-tree, which provides optimal performance when implemented on-chip. That encourages us to find out how to achieve optimal performance with fat-tree in data centers. Open queuing systems analysis followed by simulations was performed to prove the findings of the study.

This thesis contributes to the area of data center networks with the goal of finding new research directions to optimize these networks.
8.2 Future Work

In this thesis, we reviewed different DCNs topologies followed by a detailed study on DCNs problems and bottlenecks from different aspects. We gained insight into a different way of designing optimized DCNs. This analysis identified major deficiencies that should be resolved. Therefore, we are able to draw a road map for our future work, which include developing distributed adaptive routing technique that rout the traffic evenly based on the status of each link in the topology. More advanced congestion avoidance technique along with the routing technique is essential to avoid problems related to TCP. Another area of optimization includes defining a nonblocking condition for the distributed adaptive routing technique. Beside, identifying approximated buffer capacity for switches statistically. Further analysis is needed to investigate self-similarity in DCNs and its complications and the possibility of detecting it using anomaly detection techniques.
REFERENCES


