FAULT-TOLERANT NETWORK TOPOLOGIES FOR DATACENTERS

by

Heba Mahmoud Helal Attia

A Thesis presented to the Faculty of the
American University of Sharjah
College of Engineering
In Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Computer Engineering

Sharjah, United Arab Emirates

May 2017

Approval Signatures

We, the undersigned, approve the Master's Thesis of Heba Mahmoud Helal Attia

Thesis Title: Fault-Tolerant Network Topologies for Datacenters.

Signature	**Date of Signature** (dd/mm/yyyy)
___________________________ Dr. Rana Ahmed Associate Professor, Department of Computer Science and Engineering Thesis Advisor	_______________
___________________________ Dr. Khaled El-Fakih Associate Professor, Department of Computer Science and Engineering Thesis Committee Member	_______________
___________________________ Dr. Hasan Mir Associate Professor, Department of Electrical Engineering Thesis Committee Member	_______________
___________________________ Dr. Fadi Aloul Head, Department of Computer Science and Engineering	_______________
___________________________ Dr. Mohamed El-Tarhuni Associate Dean for Graduate Affairs and Research, College of Engineering	_______________
___________________________ Dr. Richard Schoephoerster Dean, College of Engineering	_______________
___________________________ Dr. Khaled Assaleh Interim Vice Provost for Research and Graduate	_______________

Acknowledgements

I would like to express my sincere appreciation and my trustful thanks to my advisor, Dr. Rana Ahmed. Without his advice and guidance, this work would not have been completed. A true mentor, he remains a figure of inspiration and I feel fortunate to have completed my thesis with him. Although he was my only advisor, he was always available to assist and guide during this thesis.

I gratefully acknowledge the help and support provided by all the professors and lab instructors in the Computer Science and Engineering department at AUS. I also acknowledge the award of Graduate Teaching Assistantship (GTA) from AUS.

Finally, I would like to thank my family and friends for all their love, support, and patience throughout my life.

Dedication

To my family and husband

You are my life

Abstract

Data centers are an integral part of cloud computing infrastructure to support various cloud-based services such as web search, email, social networking, distributed file systems and scientific computing. Data centers provide huge computational power and storage, reliability, availability, and cost-effective solutions needed by the cloud applications. A data center network (DCN) topology connects thousands of servers within the datacenter and to the external world. The topology is vulnerable to failures due to the presence of huge number of servers, switches and links. Several data center network topologies have been proposed and implemented; however, most of them lack the ability to recover from failures. One of the biggest challenges in DCN is to provide a graceful degradation in performance in the event of a link or server failure. Fault-tolerance in a DCN topology can be provided by adding extra hardware (switches, links) or by provisioning of multiple redundant routing paths among servers. This thesis proposes two new fault-tolerant DCN topologies derived from the standard $Dcell$ topology. The proposed topologies, $Dcell - Star$ and $Dcell - Ring$, are both cost-effective and scalable. In addition, the proposed topologies enhance the overall performance (throughput and latency) of $Dcell$ topology, and offer graceful performance degradation in the case of a link or server failure**.** Furthermore, we propose a new mechanism to select the optimal path between the hosts in the topology using Genetic Algorithm (GA). Performance evaluation of the proposed topologies and techniques is done through a simulation study using realistic intra-datacenter traffic models, and the results are compared with the standard $Dcell$ topology. The comparison is done in terms of various metrics such as throughput, latency, diameter, and average shortest path length. The simulation results show that the proposed topologies outperform the standard $Dcell$ topology due to the availability of multiple alternate shortest paths between any pair of servers, resulting in an improvement of about 5% in throughput even for a small-size network. GA algorithm for the path selection is applied to the two proposed topologies, and it is found that there is a further improvement of about 2% in the throughput of the topologies.

Search Terms: ***Data Center Networks, Fault-Tolerance, Throughput, Network Topology, Performance Evaluation, Mininet, $Dcell$, Genetic Algorithm.***

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABT	Aggregate Bottleneck Throughput
APL	Average Path Length
CDN	Component Decomposition Number
DCN	Data Center Network
ECMP	Equal Cost Multi-Path
FAR	Fault-Avoidance Routing
GA	Genetic Algorithm
MTU	Maximum Transmission Unit
NIC	Network Interface Card
QOS	Quality of Service
RFR	Routing Failure Rate
RWS	Roulette Wheel Selection
SCS/LCS	Smallest/Largest Component Size
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TOR	Top of Rack
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VM	Virtual Machine

Chapter 1. Introduction

In this chapter, we provide a brief introduction about datacenter network (DCN) topologies and the problems encountered in this field. We then present the scope of the problem investigated in this study as well as the thesis contribution. Finally, general organization of the thesis is presented.

1.1. Overview

Due to tremendous growth in cloud-based services, such as Google Search and Facebook social network, the overall cloud infrastructure is constantly evolving to support a huge volume of data and Quality of Service (QoS) requirements of such services. DCN infrastructure consists of a massive number of servers with large online services (e.g., Amazon, Microsoft, etc.). A DCN topology connects those servers through switches and links with high capabilities which, in turn, connects to the external world. As the result, data traffic inside the DCN is increasing, which has a strong effect on the performance of the datacenter in general. The performance of DCN can be estimated using well-known metrics like throughput, latency, bandwidth, power consumption, reliability, cost, etc. A number of DCN architectures have been proposed, which are classified into two major categories: fixed-topology, and flexible-topology. For the fixed-topology architecture, the network topology cannot be changed after the network is deployed, while the change is possible in the flexible-topology architectures. Some example architectures for fixed-topologies are: Fat-Tree [1], Portland [2], and recursive topologies $Dcell$ [3] and $Bcube$ [4]. The flexible topologies, using optical switching technology, include c-Through [5]. There is another way of classification of DCN topologies based on whether they are server-centric networks or switch-centric networks. In server-centric network, the addressing and routing tasks are performed by the servers. Both $Dcell$ and $Bcube$ topologies are examples of server-centric networks. On the other hand, all routing and addressing tasks are done by switches and routers in case of switch-centric networks. Fat-Tree and Portland architectures belong to this category.

Several requirements are needed to maintain the scalability, fault-tolerance, efficiency and management inside DCN topologies [6]. Some of the requirements are summarized as follows:

- The detection and recovery of failures should be efficient, easy and rapid.
- There are no traffic loops especially in multi-rooted architecture.
- Multiple paths are available between any two hosts.
- The ability of Virtual Machine (VM) migration without any need to change the IP address.

With ever increasing data traffic in clouds, recently there has been a great interest in improving the performance of cloud computing infrastructure. Data centers provide solutions for large power consumption and storage, cost-effective, availability and reliability which are needed by the cloud applications. There are several design and operational aspects important for selecting a network topology depending on the needed performance objectives. As a data center network topology consists of a very large number of servers (hosts), switches, routers and communication links, the topology is vulnerable to failures (permanent or transient). In order to provide high availability and a certain performance level, the network topology must be designed to recover from such failures. At the physical hardware level, redundant links and switches present in the topology may aid in providing such recovery (or fault-tolerance). Moreover, routing algorithm for the topology may offer alternate paths (if they exist) between hosts (servers) that will be available in case of a switch or link or server failure.

1.2. Thesis Objectives

Due to increased traffic in the DCN topologies, there is a strong need to improve the performance of DCN. In this work, we propose two new datacenter network architectures which have the ability to tackle the problems of failures, latency, fault-tolerance and throughput. We then compare these new proposed topologies with well-known topologies to show the improvements in the performance. These architectures provide multiple paths between hosts, achieve high throughput, and minimize the delay between the hosts. They also offer fault tolerance capability which will result from either the redundancy of physical links or the routing protocol used. The architectures are simulated and results are presented under various realistic traffic scenarios. Furthermore, Genetic Algorithm (GA) is used as a technique to select the optimal path between the nodes. The objective of this work is to find a way to improve the performance of architecture by providing different paths between hosts and then selecting the best path for routing the packets among the servers.

1.3. Research Contribution

The contributions of this research work can be summarized as follows:

- Propose new DCN architectures with higher throughput, low latency and better fault-tolerance and scalability by supporting multipath concept.
- Evaluate these architectures using simulation tools (e.g. Mininet) to verify the superiority of these architectures compared to the well-known ones.
- Use GA as path selection technique to compute the optimal path from one host to another host or from one host to all other hosts.
- Compare the results from Mininet simulator with the ones from GA to validate the ability of GA to produce better performance results.

1.4. Problem Statement

Cloud-based services are growing at an exponential rate. These services demand a reliable, scalable, highly available, high throughput and energy-efficient DCN infrastructure. The present well-known network topologies do not offer most of the above mentioned desirable features at the same time.

1.5. Thesis Organization

The rest of this thesis is organized as follows: Chapter 2 introduces the background about structure of different DCN topologies and different challenges in DCN architectures. Moreover, related work to this research topic is described. Two new proposed DCN topologies with some of their structural properties and new path selection technique using GA are discussed in Chapter 3. Chapter 4 introduces an overview of the Mininet simulator and different controllers used. Additionally, several realistic traffic patterns are presented. Chapter 5 provides results from the performance evaluation for the two new proposed architectures and compare with the current well-known topologies. In addition, Chapter 5 describes the performance results when the proposed fault-tolerant GA path selection algorithm is applied on different topologies, including the two proposed ones. Furthermore, Chapter 5 presents comprehensive comparison and discussion on the results. Finally, Chapter 6 concludes the thesis and outlines the future work.

Chapter 2. Background and Literature Review

In this chapter, we discuss the structure of various standard DCN topologies and compare them in terms of certain parameters related to the performance and fault-tolerance. Different challenges in DCN topologies are also presented. We, moreover, discuss a GA for solving the path selection problems in DCN topologies. Finally, we discuss the related work in this field of research and describe the different simulation tools used for measuring the performance of topology.

2.1. Data Center Network Architectures

Several DCN topologies have been proposed and implemented in real data centers. The list includes Google Fat-Tree, $Dcell$, $Bcube$, Facebook Fat-Tree, Diamond, and $BCCC$ topologies. Some important parameters used for comparing different topologies are presented as follows [6]:

- Diameter: It represents the longest of the shortest paths between two servers in the topology. A low value of diameter indicates lower transmission latency.
- Degree of the servers: It is the number of networking ports present on the servers.
- Number of wires: Total number of wires (communication links) required for constructing the topology.
- Number of servers: It demonstrates the scalability metric for different topologies.
- Node-disjoint Paths: It represents the minimum of total number of paths with no intermediate common nodes between any two hosts. It gives indication about available number of paths in case of switch failure.
- Edge-disjoint Paths: It shows the common edges of the available path between any two hosts, and it gives an indication of the number of available paths in case of a link failure.
- Average shortest path length: It indicates the shortest path between any two pairs of servers to cover all available paths in the network structure.

2.1.1. Fat-Tree topology. This topology, proposed by Leiserson [7], is based on a complete binary tree, and is considered the most popular one till now. It consists of different levels, as shown in Figure 2.1, starting from the core switches till the edge switches which directly connect to the hosts. Each $n-\text{port}$ edge switch is linked to $\frac{n}{2}$

hosts, while the remaining ports connect with the upper switches (aggregation switches). Each subset in the topology is called a pod that contains $\frac{n}{2}$ aggregation-level switches, $\frac{n}{2}$ edge-level switches and the hosts connected to the edge switches. All levels use the same kind of switches and the maximum number of hosts linked with the edge level is $\frac{n^3}{4}$. With the increase in traffic load, the topology presents the bottlenecks, especially in the low level. The topology offers multipath flows between any two hosts in network which makes it a fault-tolerant topology. The diameter for Fat-Tree topology is 6. This topology is also known as Google Fat-Tree.

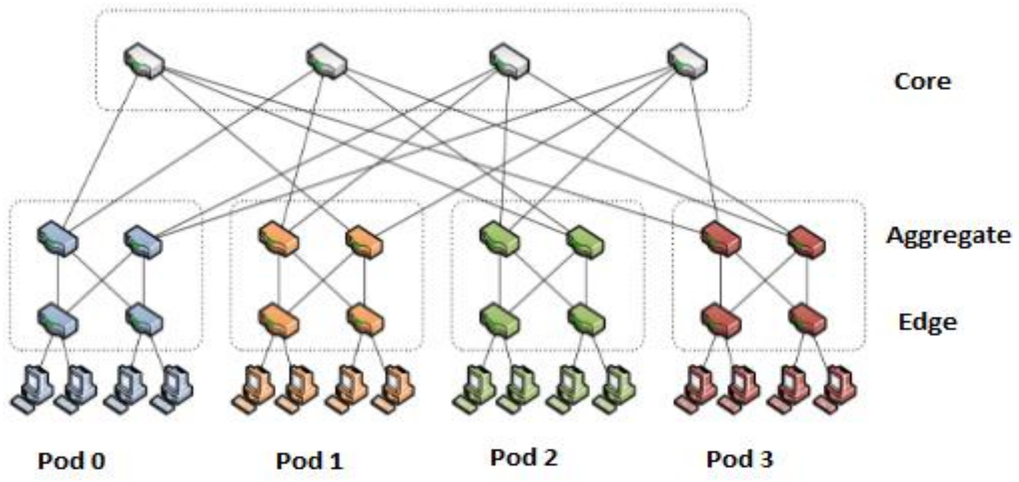


Figure 2.1: A 3-level Fat-Tree Topology [8]

2.1.2. Bcube topology. As shown in Figure 2.2, the level $-\ k\ Bcube_k$ [4] is recursively constructed from $Bcube_0$ which consists of n servers and $n-port$ switch. Here, k represents the level number in the topology. Each server is connected to one port of switch whereas there is no direct link between these servers. For the higher level, $Bcube_k$ is composed of $n\ Bcube_{k-1}$ connected together through $n^k\ n-$ port switch, in which each server port attaches to a switch at each level. This architecture utilizes the largest number of switches and wires compared to others, especially when building the higher levels. The diameter in this topology grows logarithmically with the number of servers $(log_n N)$ where n is the number of ports on a switch and N is the total number of servers in the topology.

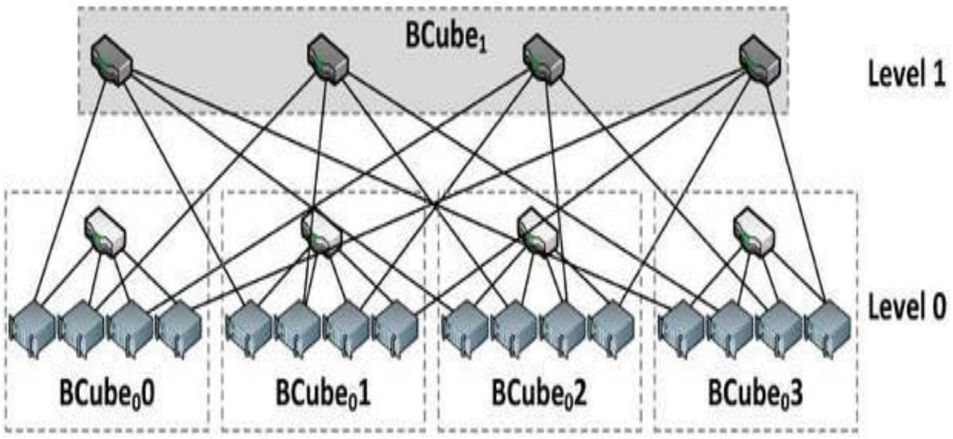


Figure 2.2: A $Bcube_1$ Topology [9]

2.1.3. Dcell topology. The $Dcell$ architecture [3] is recursively created from servers, mini-switches and links to connect servers together or servers with switches. Figure 2.3 shows the construction in general. $Dcell_0$ comprises of n servers attached to $n - port$ mini-switches. To establish the higher levels, for example, $Dcell_1$, $n + 1$ $Dcell_0$s are connected to each other via direct links. The main difference between $Dcell$ and the $Bcube$ topology is the way of scaling up, while the limiting factor of both topologies for higher levels is the number of network ports (or Network Interface Cards (NICs)) on the servers. In $Dcell$ topology, the diameter is $2^{k+1} - 1$ where k is the number of ports on a server or the level of $Dcell_k$. In addition, $Dcell$ topology is considered better scalable topology compared to the others.

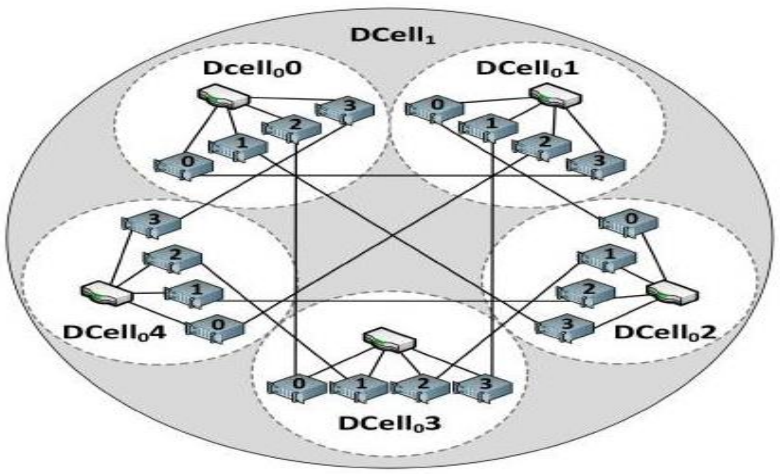


Figure 2.3: Original $Dcell_1$ Topology [9]

2.1.1. Facebook Fat-Tree topology. The standard unit of the Facebook Fat-Tree topology, as shown in Figure 2.4, is called a pod. The architecture can consist of many pods such that each pod has up to 48 switches called top of rack (TOR) switches. The architecture can be easily extended to accommodate several pods [10]. The diameter in this topology is 4.

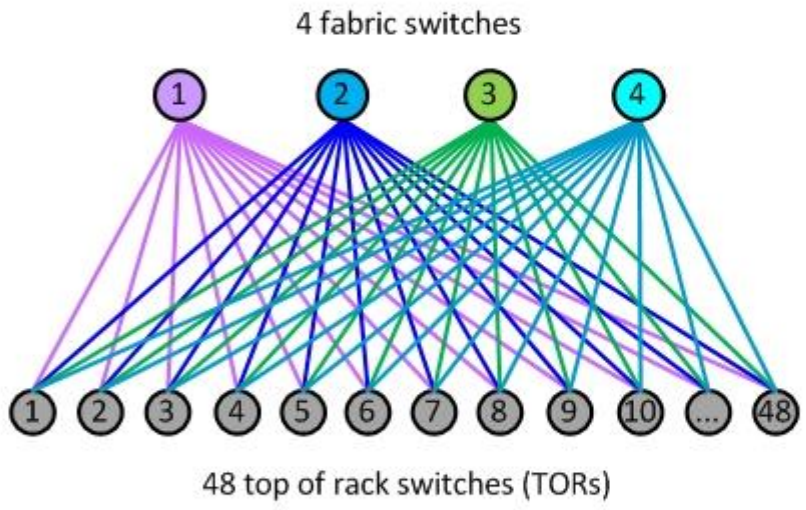


Figure 2.4: Facebook Topology [10]

2.1.2. Diamond topology. The Diamond network [11] topology, as shown in Figure 2.5, consists of only two types of switches: core and edge switches. Each edge switch has *n* ports, out of which $\frac{n}{4}$ connect to servers, $\frac{n}{4}$ connect to core switches and the remaining $\frac{n}{2}$ ports connect to the edge switches in the same pod. The network comprises of n pods, with n edge switches in each pod, with a total of n^2 edge switches in the network, $\frac{n^2}{4}$ total number of core switches, and $\frac{n^3}{4}$ total number of hosts in the network. Edge switches in the same pod as well as core switches are placed in two lines with $\frac{n}{2}$ edge switches and $\frac{n^2}{8}$ core switches in each of the upper and lower line. Each edge switch connects to the opposite edge switches in the same pod. In addition, each core switch in the upper and lower line has a link with each edge switch in the upper and lower line in each pod, respectively. In this case, the diameter is the same as in Fat-Tree topology (i.e. 6).

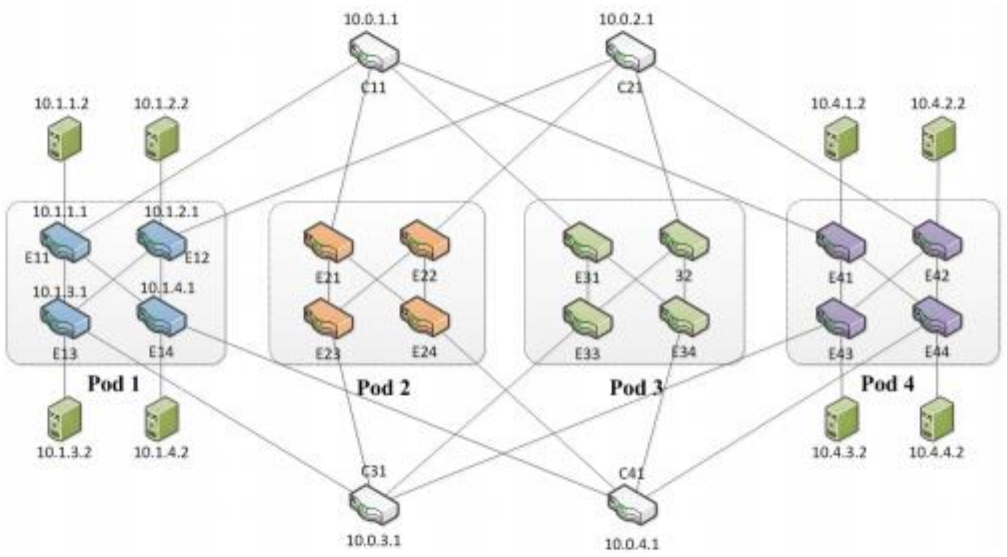


Figure 2.5: Diamond Topology with 4-port Switches [11]

2.1.3. BCCC topology. $BCCC$ topology is a recursive network structure [12], as shown in Figure 2.6, and it is derived from the $Bcube$ topology. Each element in the topology consists of n servers connected to a single $n-port$ switch. $BCCC(n,k)$ denotes $BCCC$ with order k and n number of servers which are linked to a related switch in each element. The construction of $BCCC(n,k\ )$ is done by connecting n number of $BCCC(n,k-1\ )$s, together with a total of n^k elements. In general, it requires $(k+1)n^{k+1}$ dual-port servers, $n^{k+1}(k+1)-port$ switches and $(k+1)n^k$ $n-port$ switches to build $BCCC(n,k\ )$. Moreover, there are two types of switches, called type A switch and type B switch. A type A switch is used to form an element with n ports. While a type B switch is used to connect different elements together with $(k+1)$ ports. Therefore, the first port in the server is for communication within the elements and the second one is for expansion purposes and for the communication between different elements. The diameter of $BCCC(n,k)$ is $2(k+1)$.

2.2. Challenges in DCN Topologies

There are several challenges in the present-day DCN architectures, as indicated below:

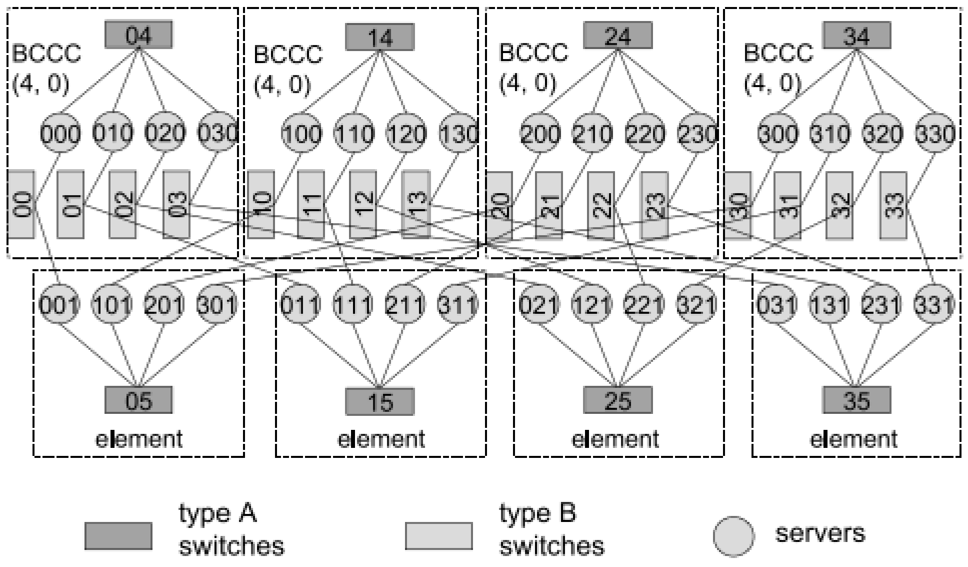


Figure 2.6: Topology of BCCC(4, 1) [12]

- Reliability: DCN hardware still suffers from frequent and disruptive failures. The number of servers, switches and links has dramatically increased in the recent past. Despite the fact that, with today's technology we have highly reliable (~99.9%) servers, links and switches, there is still a high probability for the switch or link failures. The solution for this problem is carried out by providing either a fault-tolerant routing algorithm or a topology with multiple (redundant) paths between the hosts.
- Performance: Due to strict QoS requirements of some cloud-based (especially, real-time) applications, the DCN infrastructure cannot meet the requirements of all such applications, especially under high traffic load.
- Power Consumption: Due to the presence of a huge number of servers and switches in a DCN infrastructure, the power consumption in the datacenter is very high and costly. In order to have an energy-efficient data center, one needs to design newer mechanisms to reduce to overall power consumption. One possible solution is VM migration from lightly loaded servers to other servers, and switch off those lightly loaded servers, without violating the QoS requirements of applications.

2.3. Introduction to Genetic Algorithms (GAs)

In this thesis, we present a GA for solving the path selection problems in DCN topologies. In classical GA, the search for an optimal solution for a considered problem starts by generating a population of chromosomes (possible solutions) and the evaluation of the fitness (cost function) of these chromosomes. Thereafter, an offspring population from the current one is created using the Darwinian principle of “reproduction” and “survival of fitness” [13], and the genetic reproduction operators: crossover and mutation. Reproduction involves selecting individuals from the current population based on their fitness levels. Afterwards, for each two selected individuals, crossover and mutation are applied and two new offspring individuals are created. The process is repeated for many generations until a given stopping condition is satisfied. Typically, the best individual that appeared in any generation (i.e. best-so-far individual) [14] is designated as the result produced by the GA.

2.4. Related Work

This section describes the related work reported in the literature on the subject of evaluating DCN topologies using simulation tools and the comparison among topologies using well-known metrics. Additionally, it provides various approaches on the use of GAs in finding the shortest path between two hosts in simple networks.

Several datacenter network topologies have been proposed and implemented in real data centers. The list includes Google Fat-Tree, $Dcell$, $Bcube$ and Facebook Fat-Tree. A good survey about DCN topologies can be found in [15, 16]. Most of the earlier research work carried out deals with the performance evaluation of topologies without consideration of any failure in the network topology. Moreover, different types of simulators with different traffic generators were used to evaluate the performance. Therefore, the comparisons among topologies become difficult on the similar baseline parameters. In [17], two DCN topologies, Fat-Tree and $Dcell$ topologies are compared using random traffic with exponential and uniform distribution. Simulation results from ns-3 simulator show that the performance of Fat-Tree is better than $Dcell$ in terms of throughput and latency. In [18], an evaluation of the fault-tolerance characteristics of some DCN topologies is made using a Java-based simulator. Several metrics such as Average Path Length (APL), Aggregate Bottleneck Throughput (ABT) and connection failure ratio are analyzed. A fault-tolerance parameter is measured using different metrics for four different DCN topologies (e.g. Fat-Tree, $Dcell$, $Bcube$, and $FlatNet$,) in [19]. Simulator DCNSim is used to capture the effects of various failures in the topologies in details in terms of connection-oriented metrics which include ABT, APL and Routing Failure Rate (RFR); and network size-oriented metrics, covering Component Decomposition Number (CDN) and Smallest/Largest Component Size (SCS/LCS). The performance indicates the varieties of results with respect to server-centered fault regions or switch-centered fault regions for different topologies. Both Fat-Tree and $Bcube$ topologies require higher cost for switches and wirings, although they get best ABT and APL compared to other topologies. In general, we cannot say that a certain topology is better than others, but there is always a trade-off between the performance and cost. Several popular architectures are compared with respect to various parameters, such as power, cost, scalability, hop counts and path diversity in [20] using analytical methods. It uses the "Mininet" [21] simulator for the evaluation of throughput; however, results regarding the fault-tolerance characteristics of the topologies are not considered. The work in [11] reports a new topology called Diamond, which is an improvement over the Fat-Tree topology. The Diamond topology replaces all aggregate switches with edge switches. With this new Diamond topology, the average path length and End-to-End delay decrease by 10% compared with the original Fat-Tree topology. The Diamond topology also proposes a simple routing algorithm, called Fault-Avoidance Routing (FAR), with the ability of handling thousands of servers and providing fault-tolerance. Furthermore, [12] proposes a server-centric, scalable DCN topology, named $Bcube$ Connected Crossbars (BCCC). The performance of the topology is compared with other topologies, and it is found that the BCCC topology outperforms FiConn and DCN [22] in some metrics such as network diameter, performance against server/link failure, expandability and server port utilization. In addition, the BCCC topology proposes two routing algorithms for two different types of traffic communications (i.e., One-to-One and One-to-All).

The problem of finding the optimal solution for the shortest paths between any two hosts in simple networks is well-known. GAs have been successfully applied in solving a variety of optimization problems [23, 24]. Some researchers used GAs for finding the optimal routing path between any two nodes [25] or from one node to all other nodes [26] or from all nodes to all other nodes. The work presented in [25] generates chromosomes representing routing paths or possible solutions containing different nodes between the source and destination and selects the best one after applying crossover and mutation. The resultant chromosomes have usually better routing paths than the previous population ones. Searching for better solutions (chromosomes) is done under the constraints of minimizing the path length or, in other words, the number of hop counts. The research in [26] reported similar results, but with different methods used for the representation of chromosomes and a different crossover operation. The work presented in [25-27] uses optimization strategies for high performance, scalability and optimal path for routing the packets through the network.

In [28], another technique for multipath schema in data center topologies is presented. This technique consists of two phases called the configuration phase and path selection phase. The first phase is based on a GA for generating multiple paths for minimizing the path length and increasing the link usage diversity. The proposed GA exploits the switch features in data center, which is called Virtual Local Area Network (VLAN), to map multiple paths for connecting the servers into multiple trees with the cost function of minimizing the path length and maximizing the link usage diversity. The second phase uses heuristics for making a selection between the available multi-paths. The proposed schema does not make any modification in the infrastructure while it utilizes some specifications in the data center Virtual Local Area Network (VLAN). The technique is compared with the popular multipath schema such as Spanning Tree Protocol (STP) [29], and Equal Cost Multi-Path (ECMP) [30]. The results show an increase in the transmission rates for the proposed schema even in heavy traffic (e.g. All-to-All and All-to-One).

Additionally, many recent advances in improving the performance of DCN topologies, as in Ethernet [31], are by providing the shortest paths between all pairs of hosts. By tuning the weight of the links between hosts, we can minimize the traffic delay and congestion, and also improve the throughput achieved between nodes. When any failure in link or nodes occurs, it will propagate through the network topology by the shortest path routing protocol and recalculate other shortest paths among the hosts.

Chapter 3. Proposed DCN Topologies and GA Path Selection Algorithm

In this chapter, we introduce two proposed DCN topologies derived from $Dcell$ topology. In addition, we provide some structural parameters for the topologies which are compared with the standard $Dcell$. Finally, we present detailed approaches for GAs in selecting the optimal path between the hosts in DCN topologies.

3.1. Proposed Data Centre Network Topologies

Here, we propose two new DCN topologies that are derived from the standard $Dcell$ topology with slight modifications. We have added some extra switches and links aiming to improve the performance in terms of throughput, latency and fault-tolerance. We discuss how to construct new topologies recursively.

3.1.1. $\mathbf{Dcell - Star}$ topology. Figure 3.1 shows the structure of the first proposed topology called $Dcell - Star$. In this topology, an additional (central) switch is added at the higher level to connect all mini-switches present at lower levels together. The main reasons of providing the central switch are to provide multiple alternate paths between hosts and to reduce the diameter and the average shortest path length. Figure 3.1 depicts an example of $Dcell - Star_1$ structure which consists of five basic $Dcell - Star_0$ units. Each $Dcell - Star_0$ consists of $n = 4$ servers and a mini-switch connecting them together. For constructing $\text{level} - 1$ of $Dcell - Star$ (i.e., $Dcell - Star_1$), $n + 1$ $Dcell - Star_0$ are connected through dual-port servers and the central switch joining all mini-switches. There is an increase in the number of ports for a mini-switch which needs one more port for its connection with the central switch, resulting in a total of $n + 1$ ports. The central switch also requires $n + 2$ ports; $n + 1$ ports are needed to connect all mini-switches and one port for the connection to the higher level central switch. Figure 3.2 exhibits the structure of $Dcell - Star_2$ following the similar steps in constructing topology using $Dcell - Star_1$ units with $n = 2$.

The address of a server in $Dcell - Star_k$ can be denoted as $a_t \ a_{t-1} \ \ldots\ldots\ldots a_1 a_0$ where a_0 refers to the index of the server in $Dcell - Star_0$ and it takes values from 0 to $n - 1$, and a_i represents unique ID for $Dcell - Star_0, 1 \leq i \leq k$. We can further say that two hosts $A = a_t \ a_{t-1} \ \ldots\ldots\ldots a_1 a_0$ and $B = b_t \ b_{t-1} \ \ldots\ldots\ldots b_1 b_0$ are in the same lowest level $Dcell - Star_0$ when $a_i = b_i$ and $a_0 \neq b_0$, for all values of $i, 1 \leq i \leq k$. Following a similar approach, the address of a mini-switch can be represented as $s_t\ s_{t-1}\ \ldots\ldots\ldots s_1 s_0$ where $s_0 = k - 1$ indicates the next lower level of $Dcell - Star_k$ and s_i, $1 \leq i \leq k$,represents unique ID for $Dcell - Star_0$. The addressing of the central switch is exactly the same as used with a mini-switch, except $s_0 = k$ represents the level of $Dcell - Star_k$.

As discussed earlier, two servers are in the same $Dcell - Star_k$ when all digits are equal except the least significant one. Therefore, forwarding the packets between two servers in the same $Dcell - Star_0$ is done through the related mini-switch, with $s_i = a_i$. By considering one hop count between any two directly connected devices, the hop count between two hosts in the same $Dcell - Star_0$ will be two. However, when two servers are located in different $Dcell - Star_0$ s, packets are first routed to the related mini-switch, then to the central switch which, in turn, directs the packet to the mini-switch connected to the destination node. This path needs a hop count of four to reach to the desired destination server.

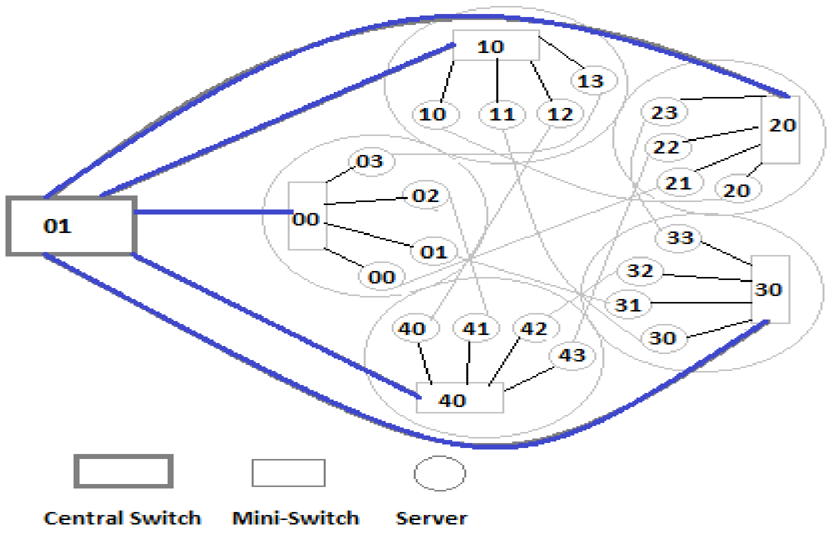


Figure 3.1: $Dcell - Star_1$ Topology with $n = 4$

3.1.1.1 Structural properties of $Dcell - Star$ topology. We analyze some vital properties of the proposed topology $Dcell - Star$. The first property is about the network diameter which gives an indication of the latency between the hosts in the topology. This property is presented in the following theorem.

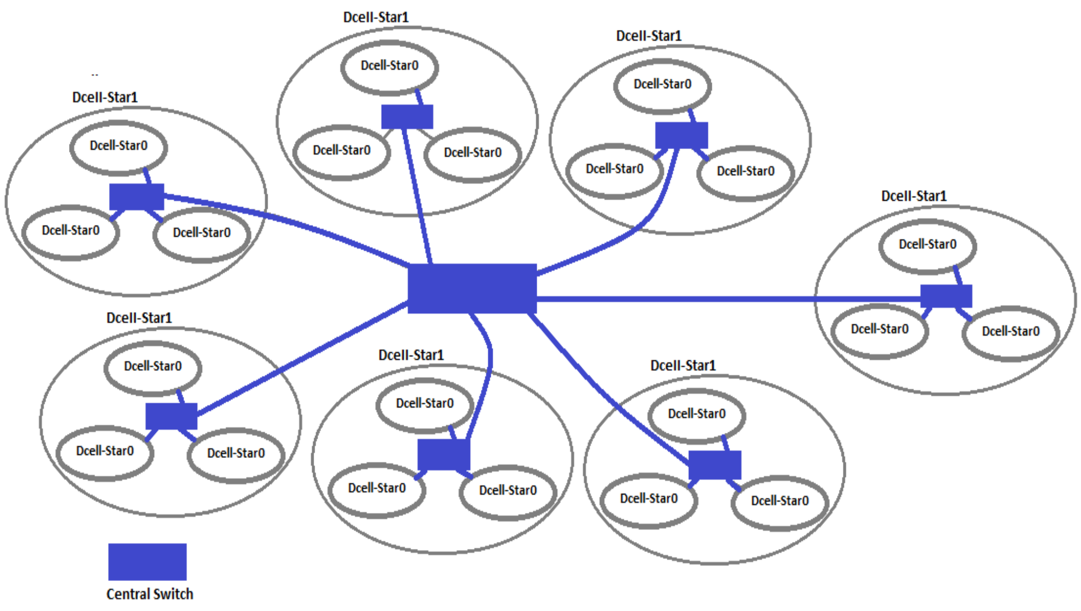


Figure 3.2: $Dcell - Star_2$ Topology with $n = 2$

Theorem 1.1: The diameter of $Dcell - Star_k$ is $2k + 2$.

Proof:

In general, moving the packets between any two neighbor servers $A = a_t\ a_{t-1}$ ……….$a_1 a_0$ and $B = b_t\ b_{t-1}$ ……….$b_1 b_0$ requires only two hop counts, since the path for the packet will be through the related mini-switch. This estimate assumes a hop count of one for the link between any two directly connected networking devices. For $level - 1$ of $Dcell - Star$, the packets are transmitted between two servers in different $Dcell - Star_0 s$ through the central switch requiring two hop counts between the related mini-switches. Therefore, for $Dcell - Star_1$, the diameter will be 2 + 2 = 4. Moving to $Dcell - Star_2$, the packets need at least 4 hop counts to reach to the desired mini-switch connected to the destination server; thus, resulting in a diameter of 6. This means that there is a relationship between the path length of the packet and the level of $Dcell - Star$. Thus, for $Dcell - Star_k$, the diameter is $2k + 2$. The second term in the diameter (2 hop counts) is because of routing the packet from the source node to its related mini-switch, and from the destination mini-switch to the destination server. Q.E.D.

Theorem 1.2: The average shortest path length between any two pairs of servers in One-to-All traffic pattern when $k = 1$ is

$$\frac{1+8(n-1)+4(n-1)^2}{n(n+1)-1} \tag{1}$$

where n is the number of ports in a mini-switch.

Proof:

For calculating the shortest path from one to all other servers, we have to consider each case one by one. For the first case, when we have a direct link between any two servers, the shortest path from the source to the directly connected server will be one hop count. In the second case, the hop count of 3 is required to reach to each neighbor of the directly connected server with a total of $n-1$ servers. In the third case, only 2 hop counts are needed for passing traffic to $n-1$ neighboring servers of the source in the same $Dcell-Star_0$. In the fourth case, 3 hop counts are required to reach to $n-1$ servers which have the direct connection with the neighbors of the source server in different $Dcell-Star_0$. For the last case, the remaining hosts need 4 hop counts when the traffic is transmitted through the central switch. Furthermore, for the last term, we have $n-1$ remaining $Dcell-Star_0s$, each with $n-1$ servers, after excluding the one covered in the previous case. Q.E.D.

Thus the total shortest path length from the source to all other hosts is:

$$SPL = 1 + 3(n-1) + 2(n-1) + 3(n-1) + 4(n-1)(n-1) \tag{2}$$

$$= 1 + 4(n-1)(n+1)$$

Then the average shortest path length S_{avg} is

$$S_{avg} = \frac{SPL}{n(n+1)-1}$$

$$= \frac{1+8(n-1)+4(n-1)^2}{n(n+1)-1}$$

$$= \frac{1+4(n-1)(n+1)}{n(n+1)-1}$$

where $n(n+1)-1$ is the total number of paths from one host to all other hosts in $Dcell-Star_1$. In the other words, $n(n+1)-1$ represents the total number of nodes in the topology excluding the source node. Q.E.D.

3.1.2. Dcell_Ring topology. $Dcell - Ring$ is our second proposed topology that also modifies the original $Dcell$ topology. In $Dcell - Ring$, all min-switches in the same level are connected together, forming a ring as shown in Figure 3.3. The main objectives of linking the mini-switches together in the form of a ring are to provide multiple alternate paths between hosts, reduce the average shortest path length and the diameter of the standard $Dcell$ topology. Figure 3.4 displays the structure of $Dcell - Ring_2$ following the similar steps in constructing topology using $Dcell - Ring_1$ units with $n = 2$.

The addressing mechanisms for servers and mini-switches are exactly the same as formulated for $Dcell - Star$. For the routing, transmitting the packets in the same $Dcell - Ring_0$ is done via the related mini-switch, with a hop count of two. However, when the source and destination servers are in different $Dcell - Ring_0s$, the packets are first routed to the related mini-switch, then towards the desired destination mini-switch through the ring, and finally to the destination server. In this case, we need a hop count of 3 when two hosts are in two different $Dcell - Star_0s$ and their mini-switches are directly connected; otherwise, the hop count will be more than three.

3.1.2.1 Structural properties of $Dcell - Ring$ topology. We now discuss some properties in the $Dcell - Ring$ topology. The first property is the network diameter presented in the following theorem.

Theorem 2.1: The diameter of $Dcell - Ring_k$, D_k, is given by $D_k = 2D_{k-1} + 1$ where the level k starts from 2. The base cases are $D_0 = 2$, and $D_1 = \left\lfloor \frac{n+1}{2} \right\rfloor + 2$ for $n < 5$, while $D_1 = 5$ for $n \geq 5$, where n is the number of servers in $Dcell - Ring_0$.

Proof:

Calculating the diameter is based on the value of n (for $n < 5$), and whether it is even or odd. This is due to the fact that the path length between two hosts in different $Dcell - Ring_0$ depends whether the related mini-switches are connected directly or not. If the mini-switches are directly connected, the total hop count between any two hosts in different $Dcell - Ring_0$ is 3. In the case where mini-switches are not directly connected, we need at least 4 hop counts. Therefore, estimating the diameter for the higher level $Dcell - Ring_k$ depends on the diameter of $Dcell - Ring_{k-1}$ which, in turn, depends on whether the number of servers in $Dcell - Ring_0$ is even or odd.

For $n \geq 5$, the destination server present in the farthest $Dcell - Ring_0s$ can be reached with a maximum hop count of 5, either by the direct connection between the related $Dcell - Ring_0s$ via the servers, or through the connected mini switches via ring. The similar approach can be used to find the diameter of higher level $Dcell - Ring_k$.

For example, the diameter of $Dcell - Ring_2$ is calculated by estimating the shortest path between the two farthest hosts as the worst case scenario. The calculation is done by estimating the diameter of two $Dcell - Ring_1s$ plus one representing the connection between them. Therefore, $D_2 = 2D_1 + 1$. We can extend the same idea to find D_k using lower level D_{k-1}. Q.E.D.

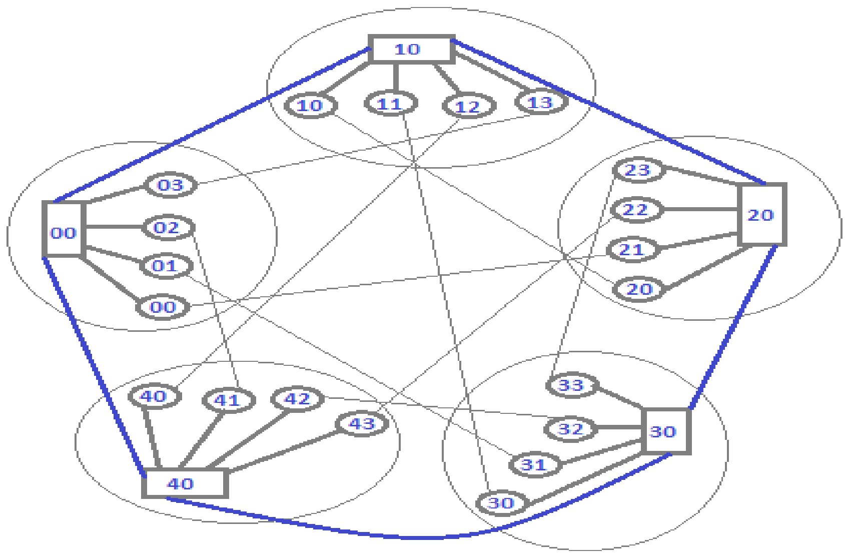


Figure 3.3: $Dcell - Ring_1$ Topology with $n = 4$

Theorem 2.2: The average shortest path length from one server to all other servers when $k = 1$ is

$$\frac{1+(n-1)(4n+3)}{n(n+1)-1}, \qquad for\, n < 5 \tag{3}$$

$$\frac{1+19(n-1)+(n-4)(5n-7)}{n(n+1)-1}, \quad for\ n \geq 5 \tag{4}$$

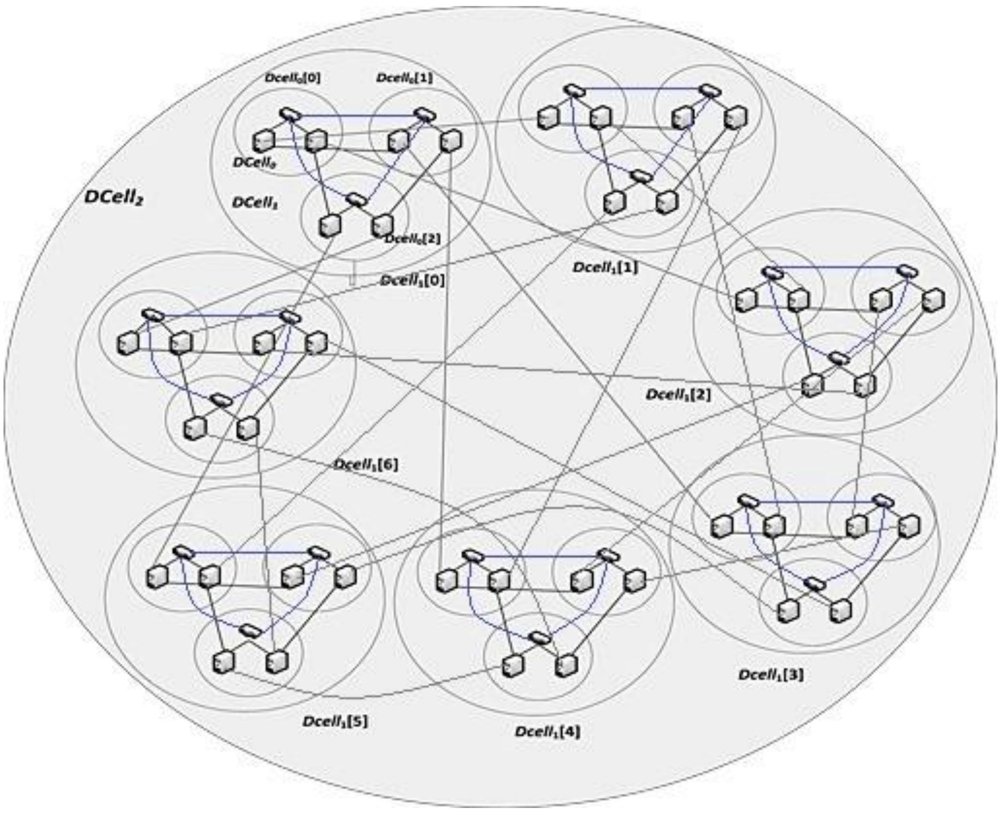


Figure 3.4: $Dcell - Ring_2$ Topology with $n = 2$

Proof:

The mechanism used for calculating the shortest path from one server to all other servers in $Dcell - Ring$ is exactly the same as used in $Dcell - Star_1$, except for the last term in equation (2). The last term in equation (2) is replaced with two terms as indicated in equation (5) for $\mathrm{n} < 5$. The first term in equation (5) represents a path length of 3 for each of $(\mathrm{n}-1)$ destination servers present in one of the immediate neighboring $Dcell - Ring_0$ connected through the mini switches. The second term in equation (5) represents a path length of 4 for each of $(\mathrm{n}-2)(\mathrm{n}-1)$ destination servers present in the equally-distant farthest $\mathrm{Dcell} - \mathrm{Ring_0}$s. Here, $(\mathrm{n} - 1)$ represents the number of servers presented in a farthest $\mathrm{Dcell} - \mathrm{Ring_0}$, and $(\mathrm{n}-2)$ indicates the number of equally distant farthest $\mathrm{Dcell} - \mathrm{Ring_0}$s.

For n ≥ 5, the formula for the total shortest path length (as given in equation (6)) is similar to the one in equation (5), except for the last term in equation (5). This last term is replaced by three terms in equation (6). Term 6 in equation (6) represents the path length of 4 of $(\mathrm{n}-1)$ destination servers present in the adjacent $\mathrm{Dcell}-\mathrm{Ring}_0$s of the immediate neighbors of the source $\mathrm{Dcell}-\mathrm{Ring}_0$. Terms 7 and 8 in equation (6) indicate the path lengths of 4 and 5, respectively, for the destination servers present in the remaining $(\mathrm{n}-4)$ $\mathrm{Dcell}-\mathrm{Ring}_0$s including the farthest ones. In term 7, the number of destination servers with path length 4 will be 2, as those two servers will have a direct connection with the server in the immediate neighbor of the source $\mathrm{Dcell}-\mathrm{Ring}_0$. In term 8, the number of remaining destination servers with path length 5 will be $(\mathrm{n}-3)$ by excluding the server with a direct connection with the neighboring servers of the source in the same $\mathrm{Dcell}-\mathrm{Ring}_0$, and by excluding two servers as indicated in term 7.

Thus the total shortest path length from a source host to all other hosts, in case of $n < 5$, is:

$$SPL = 1 + 3(n-1) + 2(n-1) + 3(n-1) + 3(n-1) + 4(n-1)(n-2) \qquad (5)$$

$$= 1 + (n-1)(4n+3)$$

Then the average shortest path length S_{avg} is

$$S_{avg} = \frac{SPL}{n(n+1)-1}$$

$$= \frac{1+(n-1)(4n+3)}{n(n+1)-1}$$

where $n(n+1)-1$ is the total number of paths from one host to other hosts in $Dcell - Ring_1$.

For $n \geq 5$, one can prove that the total shortest path length from a source host to all other hosts is:

$$SPL = 1 + 3(n-1) + 2(n-1) + 3(n-1) + 3(n-1) + 4 * 2(n-1) + 4(n-4) + 5(n-4)(n-3) \qquad (6)$$

$$= 1 + 19(n-1) + (n-4)(5n-7)$$

The average shortest path length S_{avg} is

$$S_{avg} = \frac{SPL}{n(n+1)-1}$$
$$= \frac{1+19(n-1)+(n-4)(5n-7)}{n(n+1)-1}$$

where $n(n+1)-1$ is the total number of paths from one host to all other hosts in $Dcell_1$. Q.E.D.

3.1.3. Structural Properties of Dcell Topology

Theorem 3.1: The diameter of the standard $Dcell_k$ topology is

$$D_k = 3 * 2^k - 1 \qquad (7)$$

Proof:

Since there is no direct connection in standard $Dcell$ topology among the mini-switches, the path length between any two servers will be more as compared with $Dcell - Star$ and $Dcell - Ring$. The link between any two devices in the standard $Dcell$ is considered as one hop count. Hence, the traffic communication between two servers in two different $Dcell_0s$ will be through the direct link connecting those $Dcell_0s$. For $k = 1$, the diameter will be 5 representing the longest path of all available shortest paths between any pairs of servers. This is due to the fact that the longest path occurs when we need to connect two servers present in two different $Dcell_0s$, and the direct connection between those $Dcell_0s$ is not through the source and destination servers.

We can also calculate the diameter recursively from the lower level, given as follows:

$$D_k = 2D_{k-1} + 1 \qquad (8)$$

For example, the diameter for $Dcell_2$ is estimated by calculating the diameter of two $Dcell_1$s plus one representing the direct connection between them with a total of $5 + 5 + 1 = 11$. This represents the path length between two farthest servers, as the worst case. Similarly, one can prove that the diameter of $Dcell_k$ will be two times the diameter of $Dcell_{k-1}$ plus one.

From equation (7), we can mathematically prove equation (8)

$$D_{k-1} = 3 * 2^{k-1} - 1$$

$$\begin{aligned} 2D_{k-1} + 1 &= 6 * 2^{k-1} - 2 + 1 \\ &= 6 * 2^{k-1} - 1 \\ &= 3 * 2 * 2^{k-1} - 1 \\ &= 3 * 2^{k} - 1 \\ &= D_k \end{aligned}$$

Q.E.D.

Theorem 3.2: The average shortest path length from a server to all other servers for standard $Dcell_1$ is

$$\frac{1+(n-1)(12+5(n-2))}{n(n+1)-1} \tag{9}$$

Proof:

One can use a similar approach as done in $Dcell - Star$ and $Dcell -$ Ring to prove the theorem. The path length of the last term used in SPL of $Dcell - Star$ (Equation (2)) increases because there is no direct connection among mini-switches either by a direct link or an extra central switch joining them. The routing of the packets is done through the direct communication of servers in different $Dcell_0$s. The increase in path length comes from two cases. In the first case, $(n - 1)$ servers have direct connection with the neighbor of server which has direct connection with the source server, resulting in a path length of 4. In the second case, we need to connect two servers present in two different $Dcell_0$s, and the direct connection between the $Dcell_0$s is through their neighboring servers, with a path length of 5. Furthermore, for the last term in equation (10), we have $(n - 1)$ remaining $Dcell_0$s, each with $(n - 2)$ servers, after excluding the one covered in the fourth term of the equation and one from the previous case.

The total shortest path length from the source to all other hosts will be:

$$SPL = 1 + 3(n-1) + 2(n-1) + 3(n-1) + 4(n-1) + 5(n-1)(n-2) \tag{10}$$

$$= 1 + (n-1)(12 + 5(n-2))$$

The average shortest path length S_{avg} is

$$S_{avg} = \frac{SPL}{n(n+1)-1}$$

$$= \frac{1 + (n-1)(12 + 5(n-2))}{n(n+1)-1}$$

where $n(n+1)-1$ is the total number of paths from one host to all other hosts in $Dcell_1$. Q.E.D.

The average shortest path length is based on the number of ports in a mini-switch. Figure 3.5 shows the average shortest path length for $Dcell_1$, $Dcell - Ring_1$ and $Dcell - Star_1$ with different number of hosts in $level_0$. It is concluded that it is better to use $Dcell - Star$ in a large network, especially when $n \geq 6$, and $Dcell - Ring$ in a small network. When $n \geq 6$, the average shortest path of $Dcell - Ring$ exceeds that of $Dcell - Star$ but still less than that in $Dcell$.

3.2. Proposed GA Path Selection Algorithm

We present two proposed approaches for selecting the optimal path between the hosts using the GA algorithm. The steps applied in the two approaches are the same, except in choosing the initial population. We first introduce the general steps of GA path selection algorithm, and then we outline the pseudocode of two approaches.

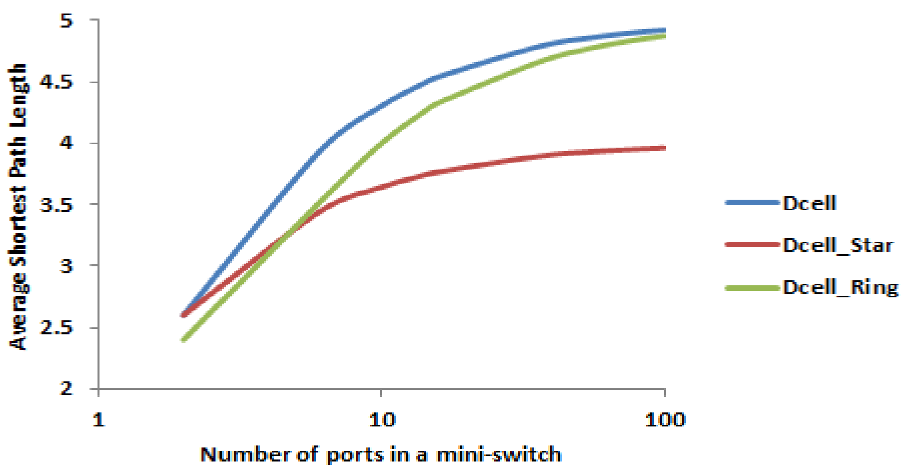


Figure 3.5: Average shortest path length vs. number of ports in a mini-switch for each topology in level $- 1$

3.2.1. Problem description. A network is represented as an undirected weighted graph, where $\mathrm{N} = \{0, \ldots., \mathrm{n}\}$ denotes the set of nodes and $\mathrm{E} = \{\mathrm{e}1, \ldots, e_m\}$ denotes the communication links between the nodes. Let $\mathrm{P} = \{n_0, n_1, n_2, \ldots \ldots, d_0\}$ represents the path from the source to the destination node, where n_0 is the source node and d_0 is the destination node. The path is chosen according to the constraint of minimizing the number of hop counts to get the shortest path between any two nodes. There are many paths available between any two nodes, especially for multipath graphs. Figure 3.6 represents Fat-Tree graph, as an example, with total number of nodes $\mathrm{N} = 36$. The bandwidth between any two nodes is represented by the weight of the link.

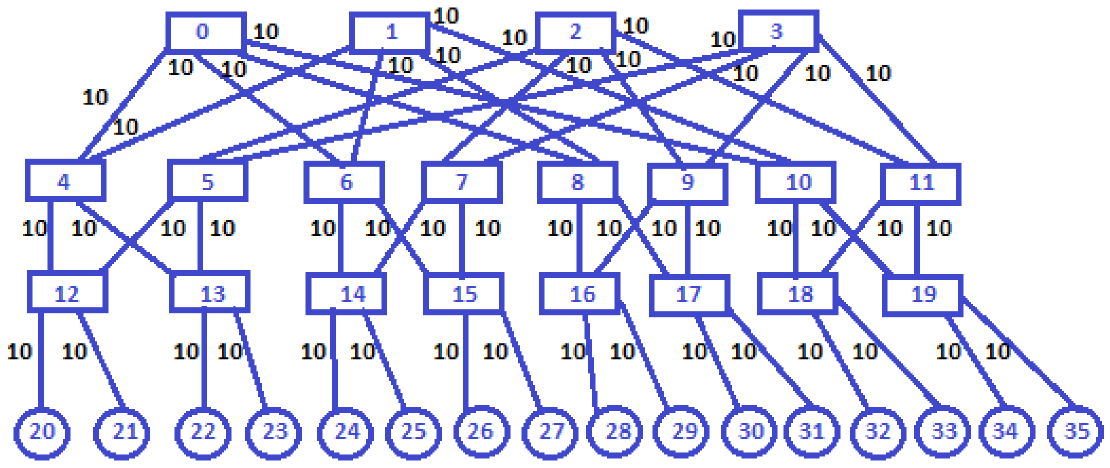


Figure 3.6: Weighted Graph of Fat-Tree Topology

3.2.2. One-to-One path selection algorithm

3.2.2.1 Chromosome representation. In our case, a chromosome is represented as a vector of integers or genes, representing a path from the given source to the given destination. Each gene in the chromosome is a node in the network topology. The first gene of a chromosome is the source node and the last one is the destination node, while the genes in between include the adjacent nodes. Chromosomes may have different lengths as there are several different length paths from the source to the destination nodes.

Assume the length of the chromosome (path) is k where the index of Gene takes values from 0 to $k-1$. The Gene $[k = 0]$ is the source node, Gene $[k-1]$ includes the destination node and Gene $[k]$ is an adjacent node of Gene $[k-1]$. As an example, as shown in Figure 3.7, the source and destination nodes are 20 and 24, respectively. Node 12 (in Gene [1]) is adjacent to node 20 (in Gene [0]). The length of the chromosome is 7.

20	12	4	1	6	14	24

Figure 3.7: A chromosome from the source node to the destination node

3.2.2.2 Fitness evaluation. The objective is to get the optimal path between the source and destination node which satisfies the following fitness (cost) function:

$Band\ (P) = \min(Hop\_Count(e), e \in E_P)$, where $Hop\_Count(e)$ represents the number of hop counts in the path and E_P refers to all available paths between any two nodes.

The fitness of each individual is evaluated by calculating the number of hop count of the path from the source to the destination nodes represented by the individual. This representation is based on the shortest path or the total bandwidth utilized between the source and destination node.

3.2.2.3 Crossover operation. The selection of individuals, from the current population, for reproduction is done using the traditional Roulette Wheel Selection (RWS) [32] as shown in Figures 3.8 and 3.9. Crossover of two selected individuals is done as follows: First, a single cut point is chosen. The position of the cut point is chosen from the range $0.3k\ to\ 0.7k$ where k represents the chromosome length, then we check all adjacent nodes at which cut point occurs in individual 1 (e.g., $Gene[4] = 6$ in individual 1). Thereafter, we scan individual 2 in reverse direction (from $Gene[k-1]$ to $Gene[0]$) to find any matching adjacent node to the node in individual 1 where the cut point is chosen (e.g., $Gene\ [7] = 14$ in individual 2). If found, we swap the part from the point where the matching adjacent node is found in individual 2 till the end of the chromosome, as highlighted in Figure 3.10, with the part in individual 1 from the chosen cut point till the end of the individual ,as highlighted in Figure 3.11. However, if no matching adjacency nodes are found, we do not apply crossover to the individual. The similar steps are applied to individual 2. The result of the crossover operation of two individuals is shown in Figures 3.10 and 3.11. In this way, we find the shortest path with high speed by reducing the length of the resulting chromosome to get the path with the least number of hop counts.

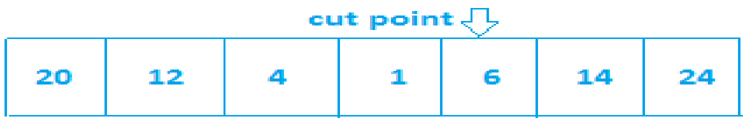


Figure 3.8: One of the selected Chromosome (Individual 1)

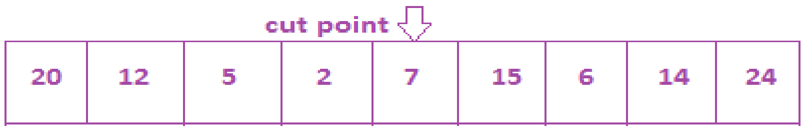


Figure 3.9: One of the selected Chromosome (Individual 2)

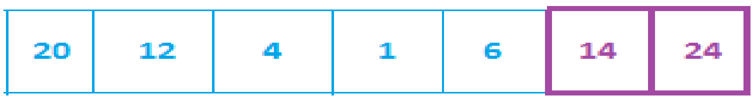


Figure 3.10: Result after crossover (Offspring 1)

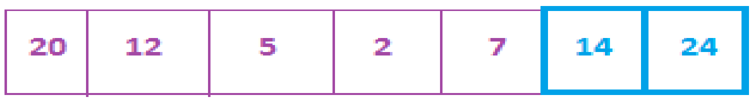


Figure 3.11: Result after crossover (Offspring 2)

3.2.2.4 **Approach$_1$.** Steps applied to find the optimal path between hosts are as the follows: First we generate initial population containing all available paths between any two hosts. Afterwards, we evaluate the fitness level of each individual in the population according to the fitness function which minimizes the hop count of the path in our case. Thereafter, we randomly select two individuals using certain selection method. After that we apply crossover operation, as discussed in details earlier. We then repeat this procedure for every two pair of individuals in the population. The result is a new population with the same size as the old one which is generated according to the fitness level. Finally, the process is repeated until certain constraints are reached which will be discussed in sub-section 3.2.5. The optimal path is the first individual selected from the new population representing the best-so-far individual. The algorithm is described in Table 3.1.

3.2.2.5 $\mathbf{Approach_2}$**.** In this approach, we apply the similar steps as described in $Approach_1$. However, after initializing the population, we only select the best individuals and apply the remaining steps on these selected individuals. This approach increases the speed of finding the optimal path and improves the quality of the obtained solution. The second approach is shown in Table 3.2.

3.2.3. One-to-All path selection algorithm. The above-mentioned GA finds the optimal path between two hosts in the One-to-One traffic model. We can extend that GA considering the One-to-All traffic model. In this case, the optimal path is determined between one (source) host and all other (destination) hosts. At the beginning, the above-mentioned GA is applied from the source node to a certain destination node as described previously using the One-to-One traffic model and get the optimal path from the source to the selected host. Using this optimal path, all adjacent nodes of the node besides the destination node (e.g. $\mathrm{Gene}\,[k-2]$) are checked. For each of these nodes, if the node is a destination node, the path from the source node to this node will be obtained directly from the obtained solution by preserving the whole path from $\mathrm{Gene}\,[0]$ to $\mathrm{Gene}\,[k-2]$ and replacing the old destination node with the new one. Otherwise, the same procedure is repeated to get the path from the source node to the remaining destination nodes.

The proposed method is fast because instead of applying GA between the source and every other destination host, the GA for finding the optimal path is only applied between the source and one of the destination host, and then by using the solution such obtained, the paths for other destination hosts can be obtained, as mentioned previously.

3.2.4. Adaptive path selection algorithm. When applying the previous algorithm to all available individuals, the obtained optimal paths will be as the same size of the initial populations. These obtained paths are ordered according to the fitness function which the first one is only picked as the best optimal path. Due to the availability of multiple optimal paths obtained between any two hosts, the proposed path selection algorithm becomes a better fault-tolerant path selection algorithm.

When any single link or switch failure occurs in the DCN topology, all these obtained optimal paths are checked, and all infeasible optimal paths are removed which are affected by the failure and then get the first feasible optimal path from the remaining ones.

3.2.5. Parameters. The GA stops searching for a solution when a termination criterion is satisfied. According to our experiments, this happens when the fitness of the best-so-far individual of all populations does not change for three consecutive iterations or when the maximum number of iterations (9) is reached. In addition, we have found that a population size (number of individuals in a population) of 10 % of all available paths between any two hosts and a crossover ratio of 0.8 yield best solutions to the considered problem.

Table 3.1: $Approach_1$, GA Path Selection Algorithm to find the optimal path from a single source to a single destination

Algorithm: Genetic algorithm for finding the shortest paths with minimum number of hop counts

```
Generation of initial population, size POP;
Evaluate fitness of the individuals;
Repeat
    Rank individuals and allocate reproduction trials;
    for (i =1 to POP step 2) do
        Randomly select 2 parents from the list of reproduction trials;
        Apply crossover and mutation;
    endfor
    Evaluate fitness of offspring;
    Preserve the fittest –so-far;
Until (best-so-far fitness is 4 times)
Solution = Fitness.
```

Table 3.2: $Approach_1$, GA Path Selection Algorithm to find the optimal path from a single source to a single destination

Algorithm: Genetic algorithm for finding the shortest paths with minimum number of hop counts

```
1. Generation of initial population, size POP;
2. Choose the best ones;
3. Evaluate fitness of the individuals;
Repeat
        Rank individuals and allocate reproduction trials;
        for (i =1 to POP step 2) do
                Randomly select 2 parents from the list of reproduction trials;
                Apply crossover and mutation;
        endfor
        Evaluate fitness of offspring;
        Preserve the fittest –so-far;
Until (best-so-far fitness is 4 times)
Solution = Fitness.
```

Chapter 4. Experimental Setup

In this chapter, we introduce an overview of the simulation tool (Mininet) used in this thesis for evaluating the performance of the DCN topologies by estimating the throughput utilization, latency and fault-tolerance capabilities of the topologies under various traffic patterns. the simulation parameters used for each considered topology are also presented.

4.1. Mininet Simulator

Mininet [21] simulator provides network analysis with a realistic virtual network of nodes, switches and links. It can create a network topology with simple shell commands (e.g., sudo mn) as shown in Figure 4.1, and a large network topology with several parameters (e.g., link bandwidth, link delay, etc.) can be programmed using python API. All network elements and flows are controlled by POX controller [33]. The controller is initialized in VirtualBox as shown in Figure 4.2. This controller helps in communicating switches to each other via OpenFlow protocol [34] which controls the forwarding of the routing tables remotely. The selection of this specific controller with respect to the other available ones (e.g. RIPLPOX, OpenDaylight, and Floodlight) is based on its demonstrated consistency in the network coverage. The traffic is transmitted from the sender to the controller which, in turn, sends it to the destination according to the information in the flow table. Moreover, STP (Spanning Tree Protocol) [29] is enabled in the network to avoid infinite loops in multi-path topologies. Figure 4.3 shows the screen snapshot of an example scenario when the Fat-Tree topology is created and the POX controller is connected.

In addition, RYU controller [35] can be used for setting static and default routes for each router in the network topology to forward the packets among the hosts. It has a capability to let a switch with a unique ID to join the topology as a router. This is beneficial especially when we want the message to follow a certain path under some required specifications like minimum hop count, low latency, shortest path, etc. Figure 4.4 and 4.5 show the screen snapshots when initializing the RYU controller and joining each switch as router, respectively.

Iperf [36] and Ping [37] commands are used for evaluating the throughput and latency, respectively, in Mininet. Iperf estimates the bandwidth consumed between any two hosts using Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), while Ping has the capability to measure the latency of packet flow, percentage of packet loss, jitter, and to specify the size of transmitted packets. All tests are run on Intel ® Core ™ i5-5200U CPU @ 2.20GHz with 6G RAM.

The Mininet simulation environment depends on the CPU and presently running applications. The Mininet emulates the behavior of a network topology; therefore, each simulation run is independent of each other.

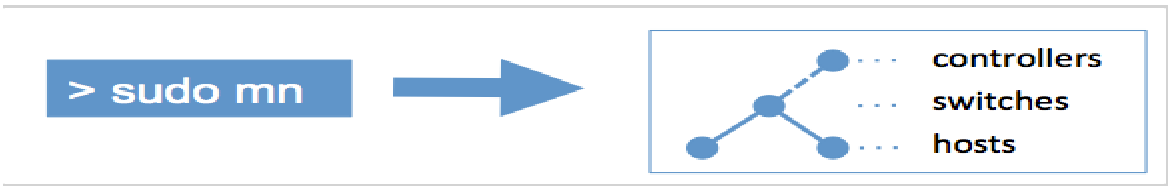
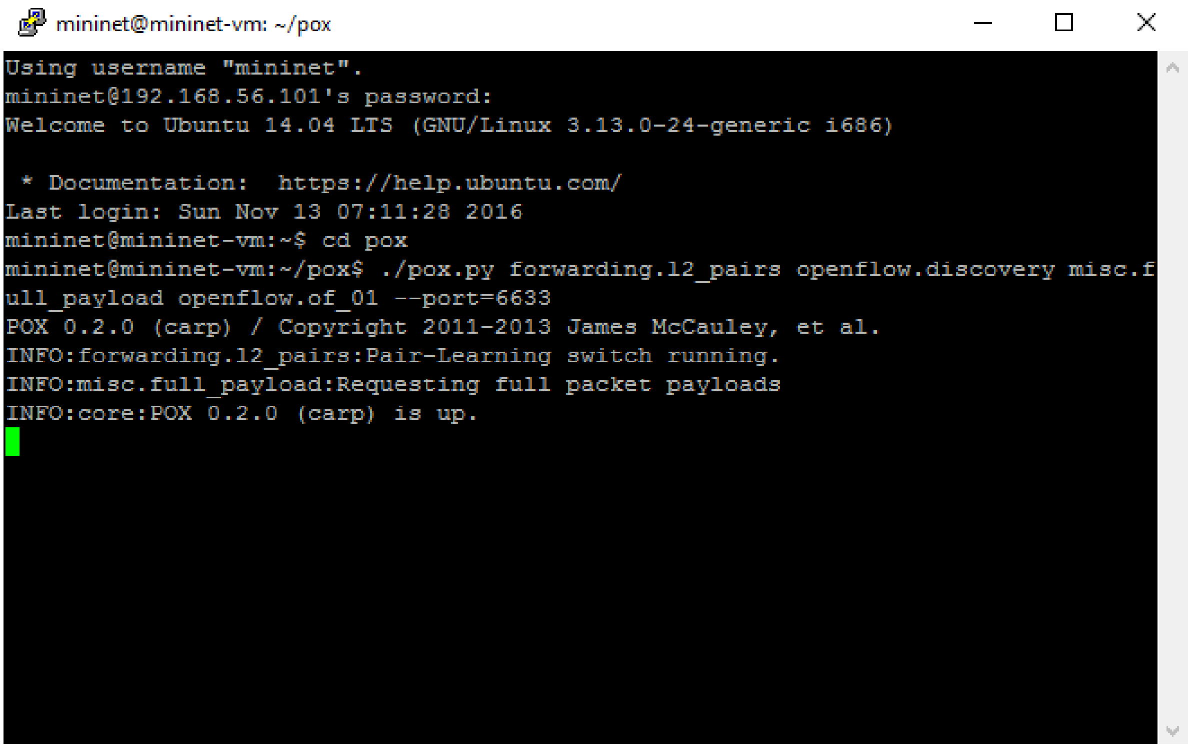
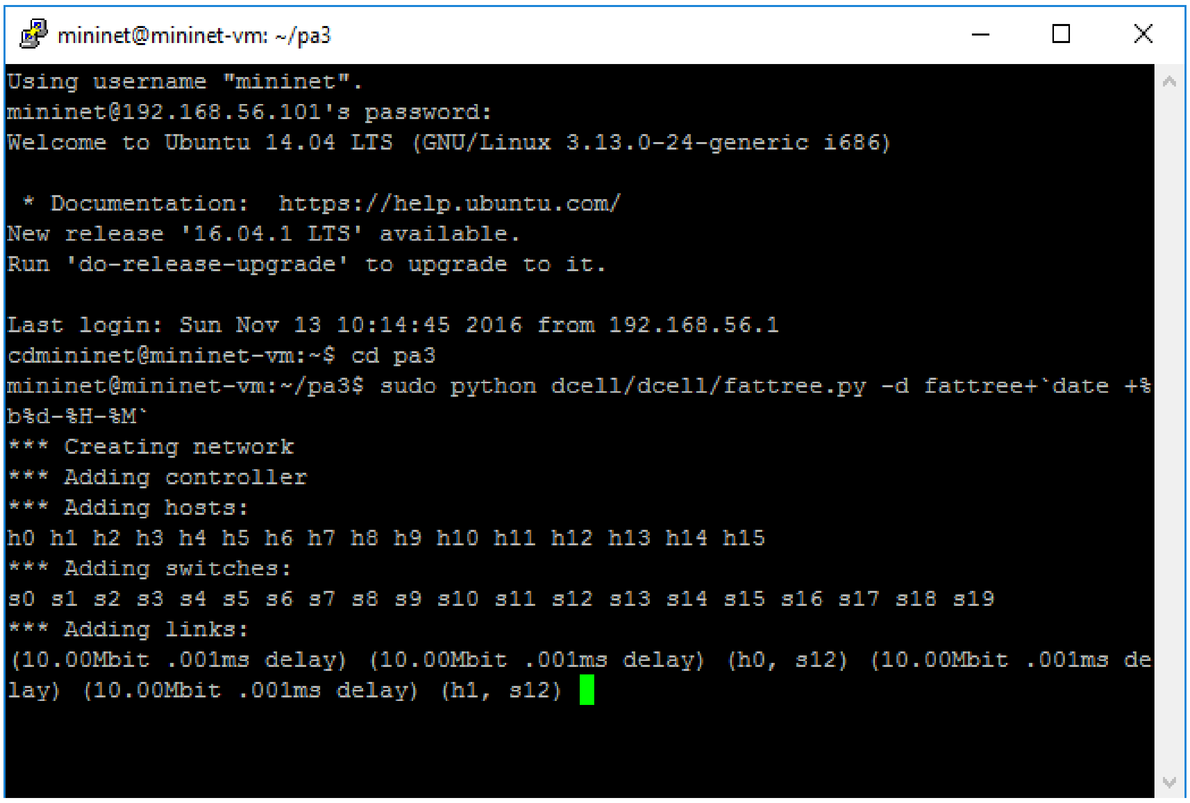


Figure 4.1: Mininet



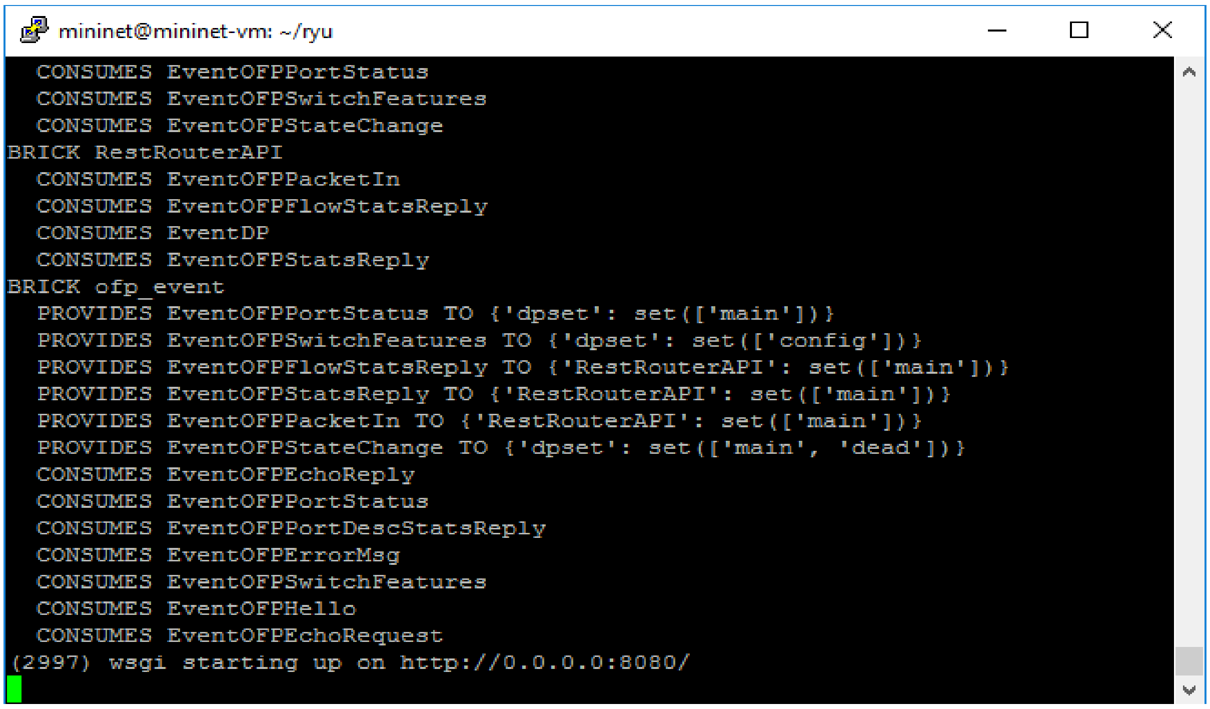
```
mininet@mininet-vm: ~/pox
Using username "mininet".
mininet@192.168.56.101's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Sun Nov 13 07:11:28 2016
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$ ./pox.py forwarding.l2_pairs openflow.discovery misc.f
ull_payload openflow.of_01 --port=6633
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:forwarding.l2_pairs:Pair-Learning switch running.
INFO:misc.full_payload:Requesting full packet payloads
INFO:core:POX 0.2.0 (carp) is up.
```

Figure 4.2: POX Controller



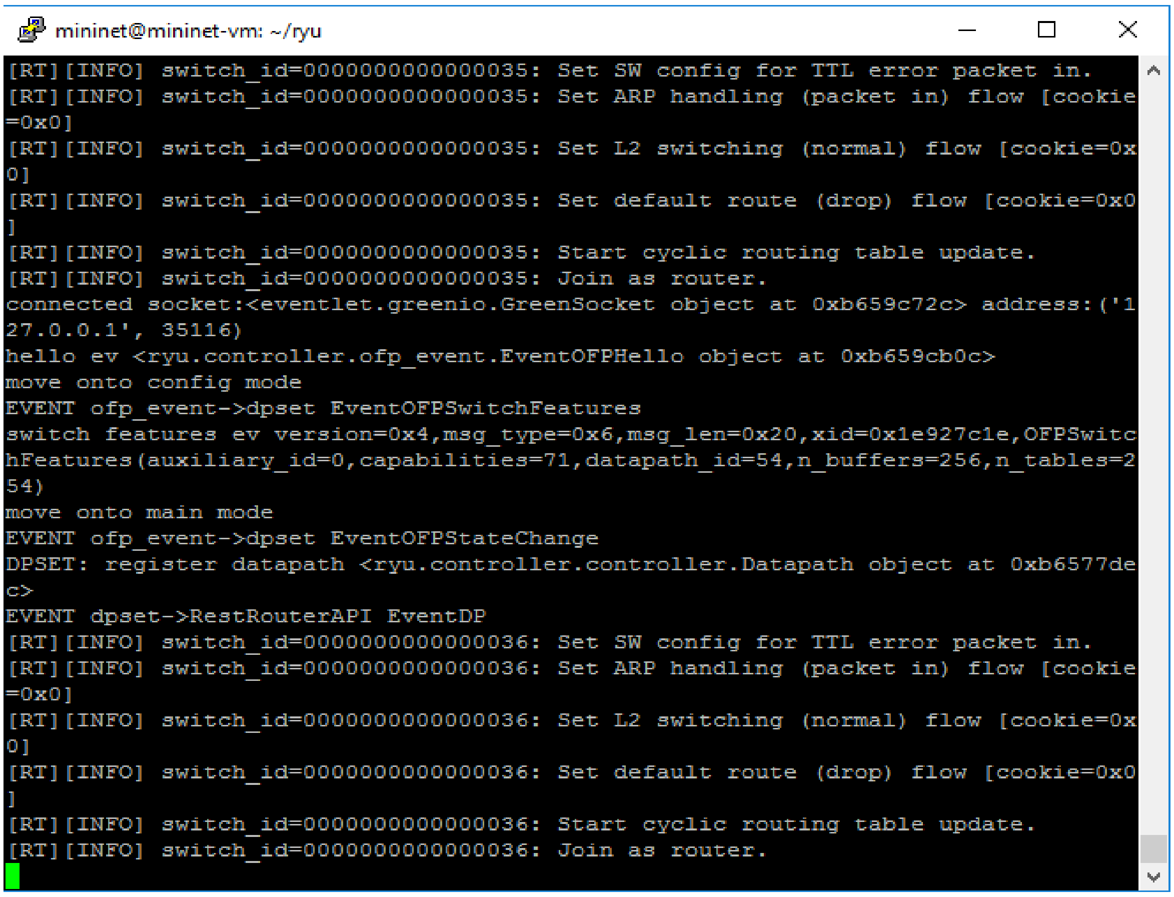
```
Using username "mininet".
mininet@192.168.56.101's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Nov 13 10:14:45 2016 from 192.168.56.1
cdmininet@mininet-vm:~$ cd pa3
mininet@mininet-vm:~/pa3$ sudo python dcell/dcell/fattree.py -d fattree+`date +%
b%d-%H-%M`
*** Creating network
*** Adding controller
*** Adding hosts:
h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Adding switches:
s0 s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19
*** Adding links:
(10.00Mbit .001ms delay) (10.00Mbit .001ms delay) (h0, s12) (10.00Mbit .001ms de
lay) (10.00Mbit .001ms delay) (h1, s12)
```

Figure 4.3: Creating Topology



```
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPStateChange
BRICK RestRouterAPI
  CONSUMES EventOFPPacketIn
  CONSUMES EventOFPFlowStatsReply
  CONSUMES EventDP
  CONSUMES EventOFPStatsReply
BRICK ofp_event
  PROVIDES EventOFPPortStatus TO {'dpset': set(['main'])}
  PROVIDES EventOFPSwitchFeatures TO {'dpset': set(['config'])}
  PROVIDES EventOFPFlowStatsReply TO {'RestRouterAPI': set(['main'])}
  PROVIDES EventOFPStatsReply TO {'RestRouterAPI': set(['main'])}
  PROVIDES EventOFPPacketIn TO {'RestRouterAPI': set(['main'])}
  PROVIDES EventOFPStateChange TO {'dpset': set(['main', 'dead'])}
  CONSUMES EventOFPEchoReply
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPHello
  CONSUMES EventOFPEchoRequest
(2997) wsgi starting up on http://0.0.0.0:8080/
```

Figure 4.4: Initializing the RYU Controller



```
[RT][INFO] switch_id=0000000000000035: Set SW config for TTL error packet in.
[RT][INFO] switch_id=0000000000000035: Set ARP handling (packet in) flow [cookie
=0x0]
[RT][INFO] switch_id=0000000000000035: Set L2 switching (normal) flow [cookie=0x
0]
[RT][INFO] switch_id=0000000000000035: Set default route (drop) flow [cookie=0x0
]
[RT][INFO] switch_id=0000000000000035: Start cyclic routing table update.
[RT][INFO] switch_id=0000000000000035: Join as router.
connected socket:<eventlet.greenio.GreenSocket object at 0xb659c72c> address:('1
27.0.0.1', 35116)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0xb659cb0c>
move onto config mode
EVENT ofp_event->dpset EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0x1e927c1e,OFPSwitc
hFeatures(auxiliary_id=0,capabilities=71,datapath_id=54,n_buffers=256,n_tables=2
54)
move onto main mode
EVENT ofp_event->dpset EventOFPStateChange
DPSET: register datapath <ryu.controller.controller.Datapath object at 0xb6577de
c>
EVENT dpset->RestRouterAPI EventDP
[RT][INFO] switch_id=0000000000000036: Set SW config for TTL error packet in.
[RT][INFO] switch_id=0000000000000036: Set ARP handling (packet in) flow [cookie
=0x0]
[RT][INFO] switch_id=0000000000000036: Set L2 switching (normal) flow [cookie=0x
0]
[RT][INFO] switch_id=0000000000000036: Set default route (drop) flow [cookie=0x0
]
[RT][INFO] switch_id=0000000000000036: Start cyclic routing table update.
[RT][INFO] switch_id=0000000000000036: Join as router.
```

Figure 4.5: Joining the switches as router in RYU Controller

4.2. Simulation Parameters

Several standard well-known DCN topologies are evaluated via simulation. The Google Fat-Tree topology used for simulation consists of three levels: core, aggregate and edge; while at the fourth level we have hosts connected. Each edge switch connects to two hosts with a total of 16 hosts. We can extend the number of hosts to any number, which is limited by the number of ports in the switch.

The $Dcell$ topology is implemented for $level - 1$ ($Dcell_1$). It consists of five switches with four hosts connected to each switch. There is one link between two hosts present in different $Dcell_0$s. We encounter a problem in forwarding the packets from a host having two different network interfaces (ports) in Mininet. Using an approach used in [38], we solve this problem by introducing a combo-switch connected to each host. This switch allows the two interfaces attached to the server to send packets simultaneously. In addition, the two proposed topologies (e.g., $Dcell - Star$ and $Dcell - Ring$) are simulated with the same set of parameters.

The $Bcube$ topology is implemented for $level - 1$ with core and edge switches. Each edge switch is attached to 4 hosts with a total 16 hosts. As each host has two interfaces, the similar packet forwarding problem mentioned above for $Dcell$ topology exists, and the problem is solved using similar approach as in [38].

The Facebook Fat-Tree topology is implemented using 4 fabric switches and 16 TORs. Each TOR switch connects to only one host, and we have a total of 16 hosts.

4.3. Traffic Patterns

Different traffic models are used to inject data traffic into the DCN topologies in order to evaluate their performance. The same traffic models described in [39] are used, and these models are representatives of intra-data center traffic scenarios. The following traffic models are used:

- Uniform Random Traffic: In this model, each host sends traffic to any random host with the same probability.
- Stride Traffic: In this model, any host with an index i send packets to a host with an index $\left(i + \frac{N-1}{2}\right) mod\ N$ where N is the total number of hosts.
- Bit Complement Traffic: Here, each node forwards the traffic to another node with an index of bitwise inversion of the sender node.
- One-to-All: Here, one host sends packets to all other hosts and the results are averaged.
- All-to-All: Here, all hosts send packets to all other hosts. This traffic model represents the highest traffic loads applied to the network.

Chapter 5. Results and Analysis

In this chapter, the simulation results achieved on well-known DCN topologies and the two proposed topologies are prsented, and then evaluate and compare different metrics (such as throughput and latency with and without failures) for each one. Finally, the GA path selection algorithm is applied on the standard and proposed topologies to measure the enhancement or the gain after applying GA and selecting the optimal path.

5.1. Comparisons among Fat-Tree, Facebook, $Bcube$, and $Dcell$

Now, a comparison among four standard DCN topologies is introduced in terms of three known metrics; throughput, latency and fault-tolerance.

5.1.1. Throughput. Figure 5.1 depicts the simulation results for the normalized throughputs of the four standard topologies under different traffic models. The normalized throughput for a certain type of traffic workload in each topology is measured by averaging over ten independent runs. From the figure, we can conclude that Facebook Topology has the highest throughput followed by Fat-Tree, $Bcube$ and $Dcell$. This is mainly due to the larger number of disjoint paths among hosts, in the Facebook and Fat-Tree topologies that could potentially reduce the traffic congestions. Moreover, both $Bcube$ and $Dcell$ get relatively close results. As expected, All-to-All traffic gets the smallest throughput compared to all other types of traffic.

5.1.2. Latency. Statistics for average packet latency for different standard topologies are gathered via simulation. We use various packet sizes for the All-to-All traffic model where all hosts send packets to every other host in the network in parallel. The results are summarized in Figure 5.2. The Maximum Transmission Unit (MTU) represents the largest packet size (1500 bytes) used, while the default value is 56 bytes. Facebook Fat-Tree topology has the lowest latency which is mainly due to its smallest diameter, despite the complexity of the topology. The latency for the Fat-Tree topology is slightly higher as compared to the Facebook Fat-Tree, due to its slightly higher diameter. Although both $Dcell$ and $Bcube$ topologies have lower diameters than the Fat-Tree topology, both of them have higher values for latency. This is due to the fact that the way we are simulating those topologies, it is not allowed to have two interfaces for a server/ host in the Mininet simulation environment. As mentioned in Section 4.2, we have added a dummy combo-switch to facilitate packet forwarding on both interfaces on a server. This dummy switch, in fact, increases the number of switches in the topology which, in turn, increases the diameter of the simulated topology as compared to the original *Dcell* and *Bcube* topologies. As a result, *Dcell* and *Bcube* topologies produce higher latencies when we increase the packet size, especially in bottleneck links and switches. Figure 5.3 shows the simulation results for the topologies for the One-to-One traffic model.

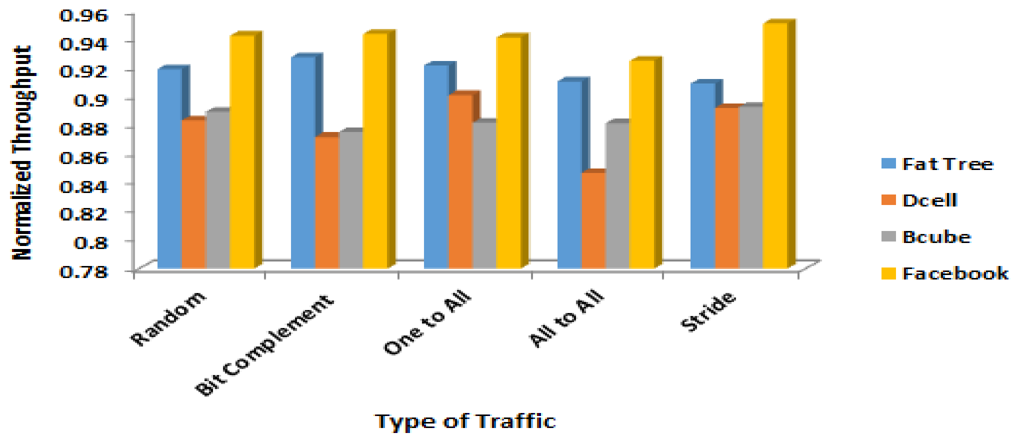


Figure 5.1: Throughput comparison for the standard topologies under different traffic types

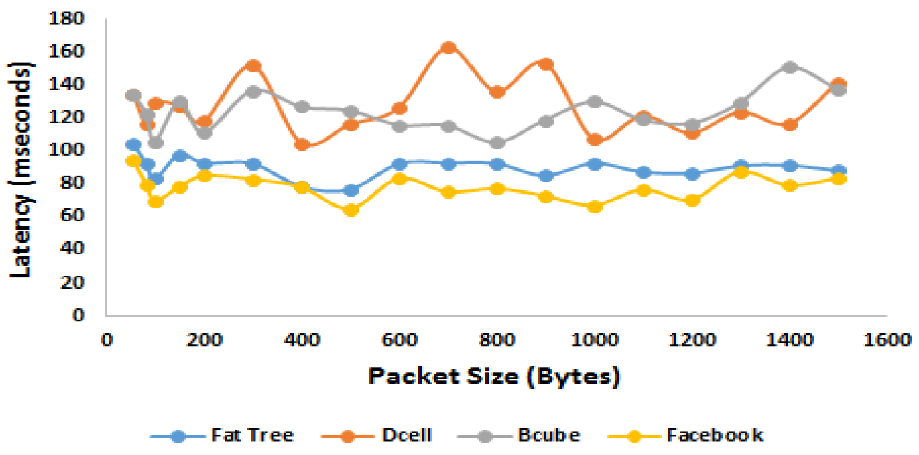


Figure 5.2 Latency comparison for the standard topologies with different packet size (All-to-All traffic)

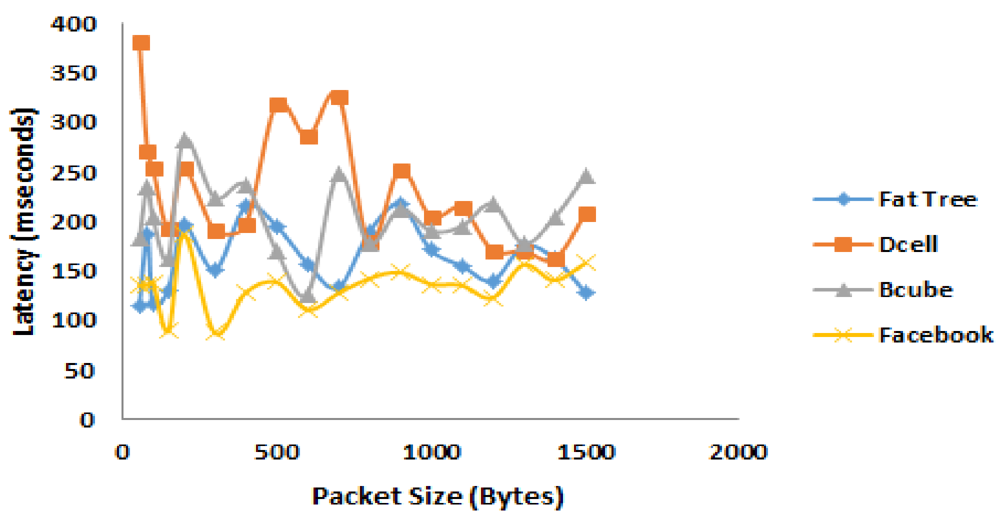


Figure 5.3: Latency comparison for the standard topologies with different packet size (One-to-One traffic)

5.1.3. Fault-Tolerance. To evaluate the effect of failures in the network topology, we simulate the topology and estimate the throughput under two types of failures: transient (failure present for a certain short fixed time period) and the permanent failure (failure lasting during the entire simulation time). We simulate a link and a switch failure. The failure region indicates the place of failure such that the failure in low level (edge level) will totally drop the packets because there is only one gateway for the packet; whereas in case of failure at upper levels, the packet may possibly find an alternative path to the destination server. Figures 5.4 and 5.5, respectively, show the behavior of *Dcell* topology with transient and permanent failure in upper level in case of One-to-One traffic between the two hosts located at the largest possible distance away in the network. The behavior is monitored at the interface (switch port) connected towards the lower level (for example, aggregation level) in the topology while the failure occurs at the link connected to a switch port towards the higher level (for example, core level). In Figure 5.4, the first drop in average throughput is due to the link failure whereas the second one is due to the switch shutdown. The transient failure is simulated to be present in the system for 5 msec. The simulation of permanent failure, as shown in Figure 5.5, indicates how the network will recover from the failure by finding out another alternate path, if it exists, for forwarding the packets. The behaviors of both *Dcell* and Fat-Tree topologies are relatively the same, except for the achievable throughput in a topology as discussed earlier. Fat-Tree topology outperforms *Dcell* topology in the throughput metric as more paths are available between any given pair of hosts. When simulating for All-to-All traffic, we find the degradation in the performance with time. Figures 5.6 and 5.7 represent the performance of the network with All-to-All traffic with large packet size (5G bytes) being transmitted for Fat-Tree and *Dcell* topologies, respectively.

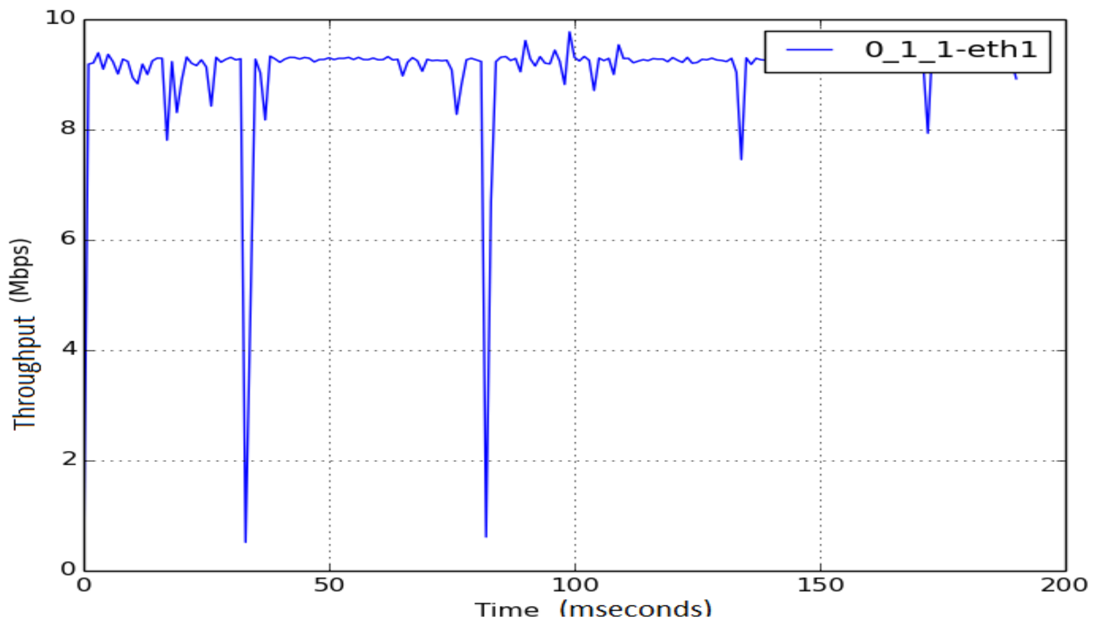


Figure 5.4: *Dcell* transient failure (One-to-One traffic)

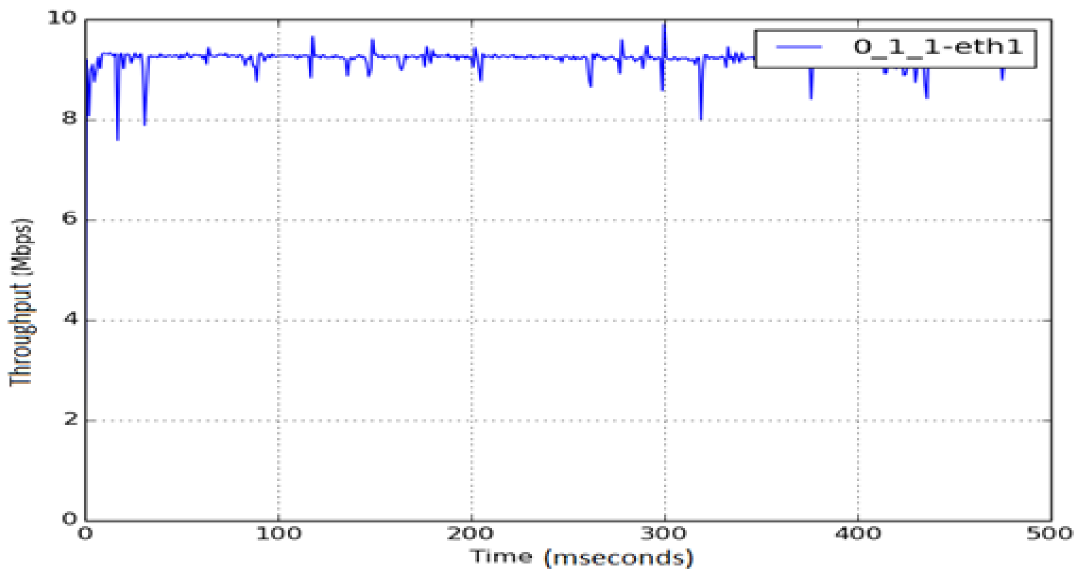


Figure 5.5: *Dcell* permanent failure (One-to-One traffic)

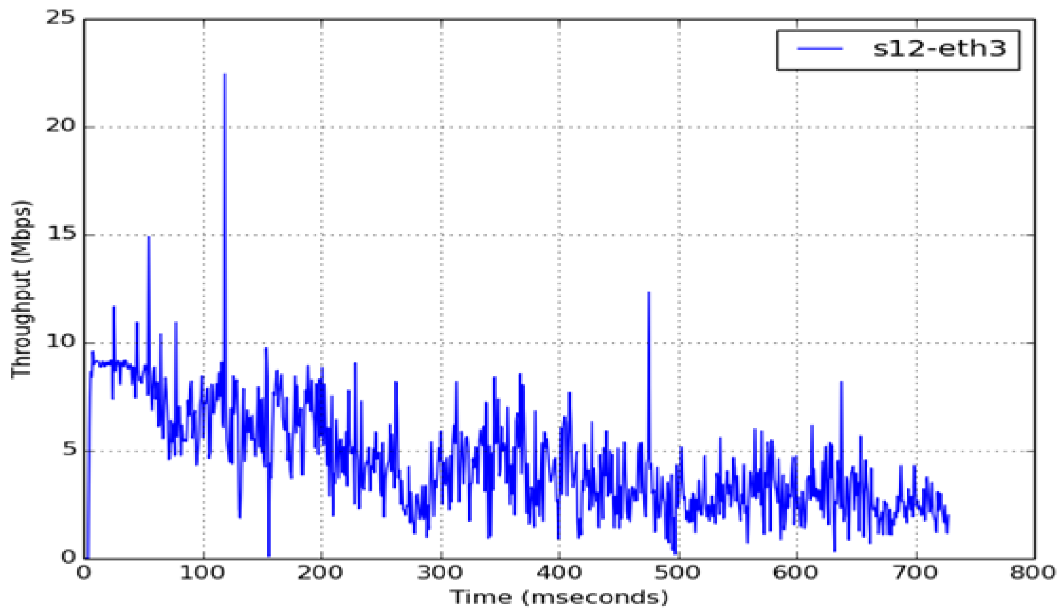


Figure 5.6: Fat-Tree topology permanent failure (All-to-All traffic)

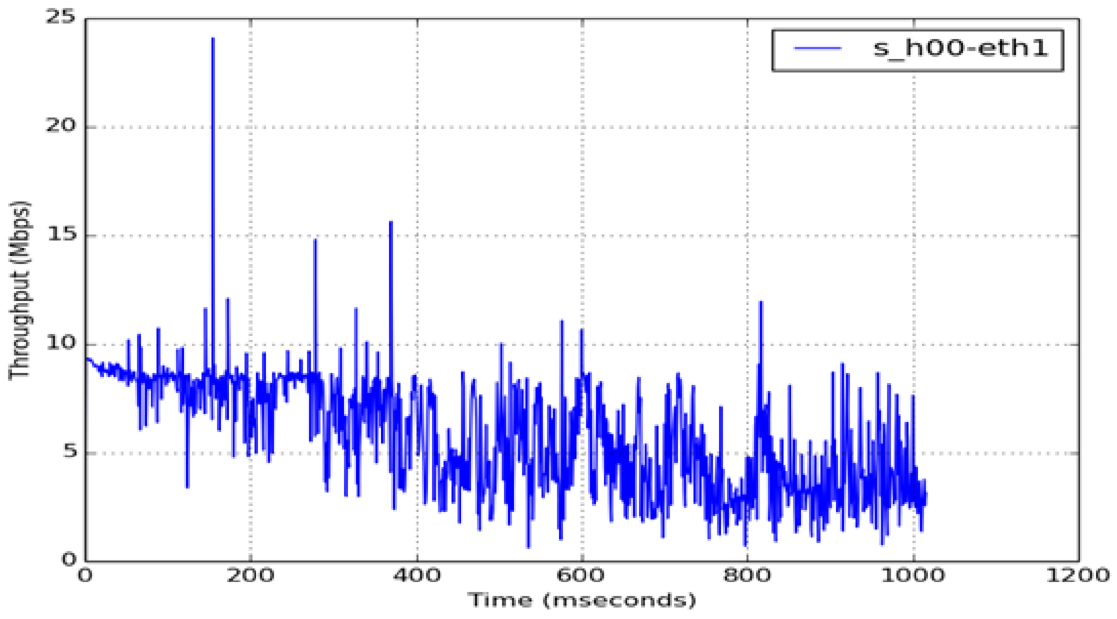


Figure 5.7: $Dcell$ topology permanent failure (All-to-All traffic)

5.2. Comparisons among $Dcell$, $Dcell - Star$ and $Dcell - Ring$

In this subsection, we compare the standard $Dcell$ topology with the two proposed topologies with varieties of parameters such as throughput, latency, fault-tolerance, diameter, and average shortest path length.

5.2.1. Throughput. Figure 5.8 shows the simulation results for the normalized throughputs of the two proposed topologies ($Dcell - Star$ and $Dcell - Ring$) and the results are compared with the original $Dcell$ topology under different traffic models. We can conclude, from the figure, that $Dcell - Star$ and $Dcell - Ring$ topologies have the higher throughput compared to the standard $Dcell$ topology. In addition, $Dcell$ and $Dcell - Ring$ topologies get relatively close results. Increase in throughput in the proposed topologies is mainly due to the larger number of the shortest paths that could potentially reduce the traffic congestions by transmitting the traffic in parallel.

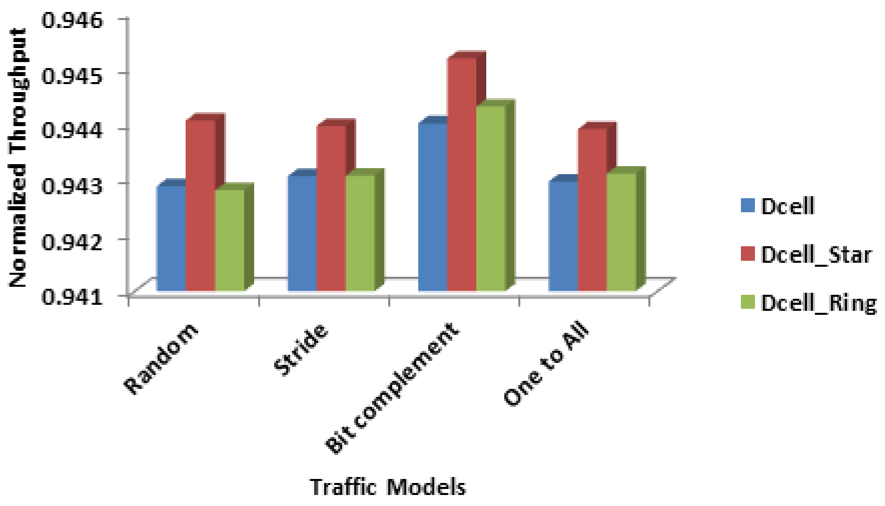


Figure 5.8: Throughput comparison for $Dcell$, $Dcell - Star$ and $Dcell - Ring$ topologies under different traffic types with $n = 4$

As increasing n, the total number of hosts in the topology increases, and we observe higher percentage improvements in the throughput results for both proposed topologies compared with the standard $Dcell$. As an example, for $n = 16$, $Dcell - Star$ and $Dcell - Ring$ exhibits an improvement of about 5% and 3% for the throughput results, respectively, in One-to-One traffic pattern. Moreover, an improvement of about 3% and 2.5% in throughput for the All-to-All traffic model is achieved for $Dcell - Star$ and $Dcell - Ring$, respectively, as compared with $Dcell$ topology. As shown in Figure 5.9, the throughput achieved in two proposed topologies exceeds that in the standard $Dcell$ topology for almost all traffic models. However, the increase in the throughput in $Dcell - Star$ is higher than the $Dcell - Ring$ for all traffic scenarios.

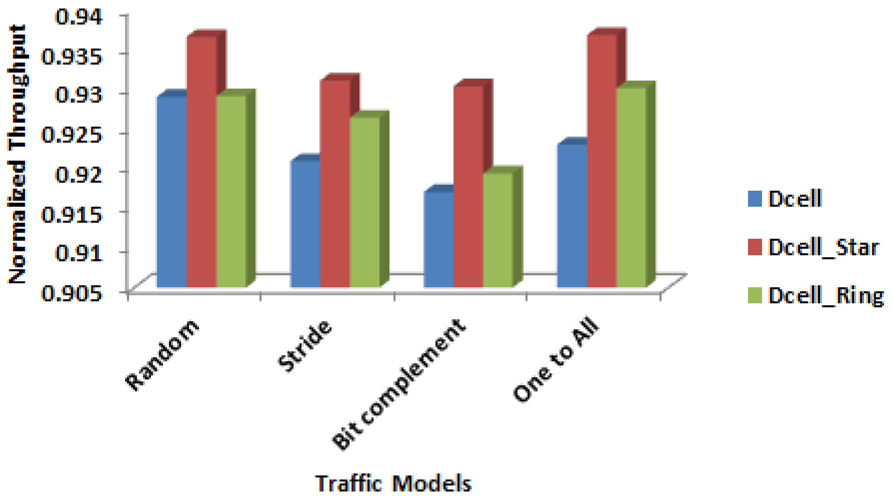


Figure 5.9: Throughput comparison for $Dcell$, $Dcell - Star$ and $Dcell - Ring$ topologies under different traffic types with $n = 9$

5.2.2. Latency. Figure 5.10 summarizes the simulation results for the latency for the standard $Dcell$ and the two proposed topologies. The proposed topologies have smaller latencies as compared to the original $Dcell$ due to their lower diameters. It is to be noted that the combo-switch approach is also used when simulating the two proposed topologies.

5.2.3. Fault-Tolerance. In order to study the effect of failures in two new proposed network topologies compared with the standard $Dcell$ topology, we simulate each topology and monitor the behavior of the topology under two types of failure: transient and permanent failure. We simulate a link and a switch failure. Due to the presence of the parallel shortest paths in $Dcell - Star$ and $Dcell - Ring$ topologies, both achieve better fault-tolerance characteristics as compared to the original $Dcell$ topology. Transient and permanent failures generate similar behavior, as shown in Figure 5.11 in case of One-to-One traffic between the two hosts located at the largest possible distance away in the network for the proposed topologies. The transient failure is simulated to be present in the system for 5 msec and the behavior is monitored at the interface (switch port) connected towards the lower level in the topology. When the controller notices any failure in the link or switch, it chooses the next shortest path available between any two hosts. The standard $Dcell$ topology gets similar results, as shown in Figure 5.12, but with a slightly lower throughput. The proposed topologies outperform $Dcell$ topology in the utilization of most links. In the standard $Dcell$, there are more ripples (sudden changes) in the throughput over time as compared to the proposed topologies, as shown in Figure 5.11.

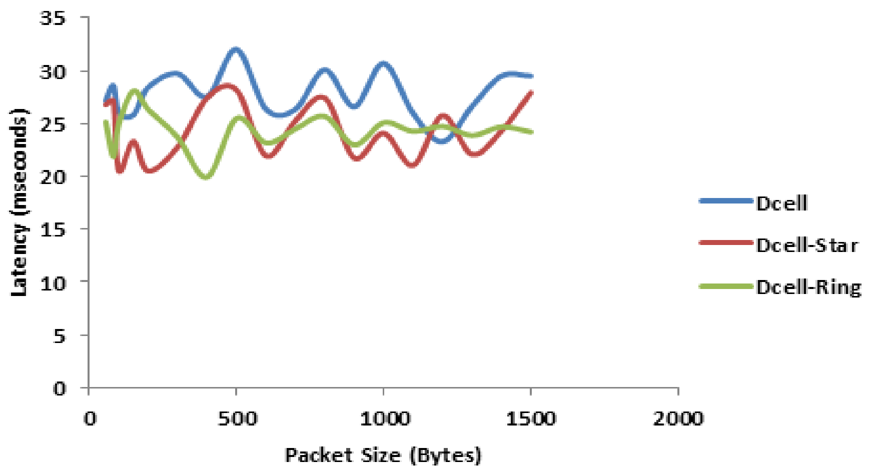


Figure 5.10: Latency comparison for $Dcell$, $Dcell - Star$ and $Dcell - Ring$ topologies with different packet sizes in All-to-All traffic model

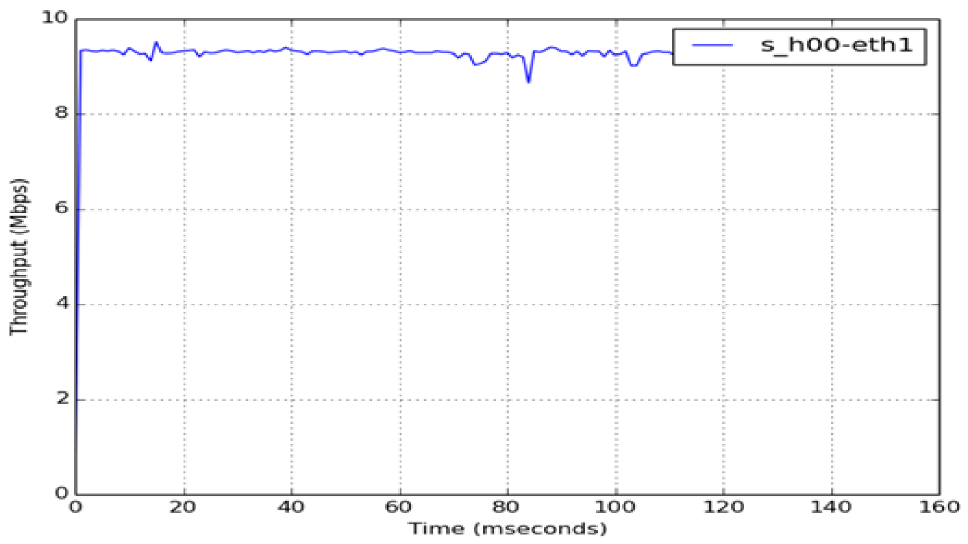


Figure 5.11: $Dcell - Ring$ and $Dcell - Star$ transient failure (One-to-One traffic)

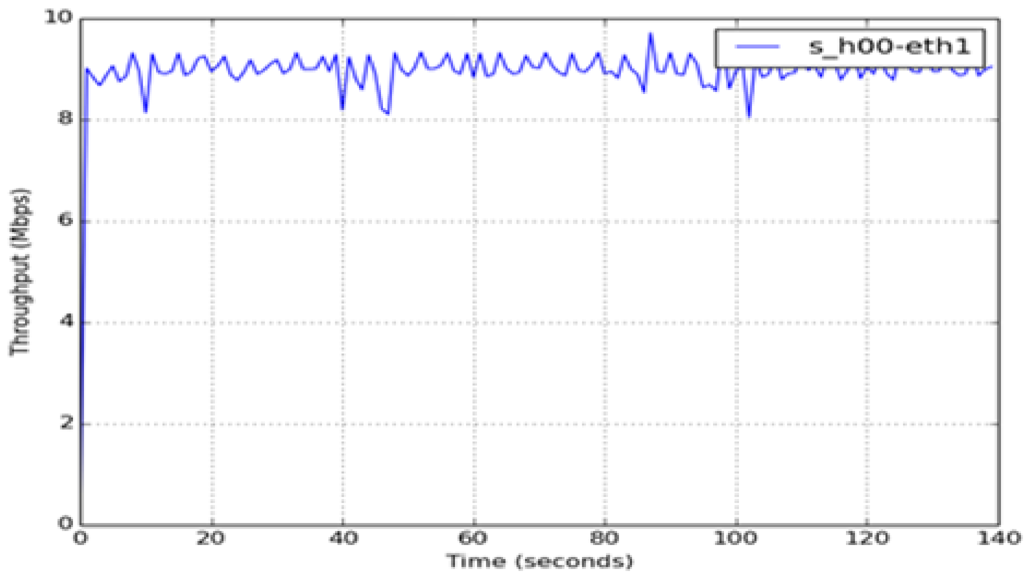


Figure 5.12: $Dcell$ transient failure

(One-to-One traffic)

5.2.4. Diameter. Figure 5.13 depicts the diameter of each topology with the same number of servers in $level - 0$ $(n = 4)$. $Dcell$ topology has the largest diameter at all topology levels followed by $Dcell - Ring$, and then $Dcell - Star$. By increasing the number of servers in $level - 0$, the diameters for both $Dcell$ and $Dcell - Star$ do not change, while the diameter of the $Dcell - Ring$ diameter increases when $n < 5$ and then becomes constant for $n > 5$, as shown in Figure 5.14.

$Dcell - Star$ shows the lowest value of the diameter among the three topologies. For example, when $k = 5$, the diameter of $Dcell - Star$ is only 12 as compared to 95 in the standard $Dcell$, and 79 in $Dcell - Ring$. Therefore, one can expect lower values for delays in routing packets in $Dcell - Star$.

5.2.5. Performance evaluation. Extensive simulations are conducted in order to evaluate the properties of average path length in case of two models of failure: random link and random server failures. In the random link failure model, there will be no packets transmitted through the affected links, while a fraction of servers is not able to transmit packets in case of random server failure model. Finally, we also evaluate the effect of failure on the network topologies when increasing the total number of hosts. The results are reported by averaging over 15 simulation runs.

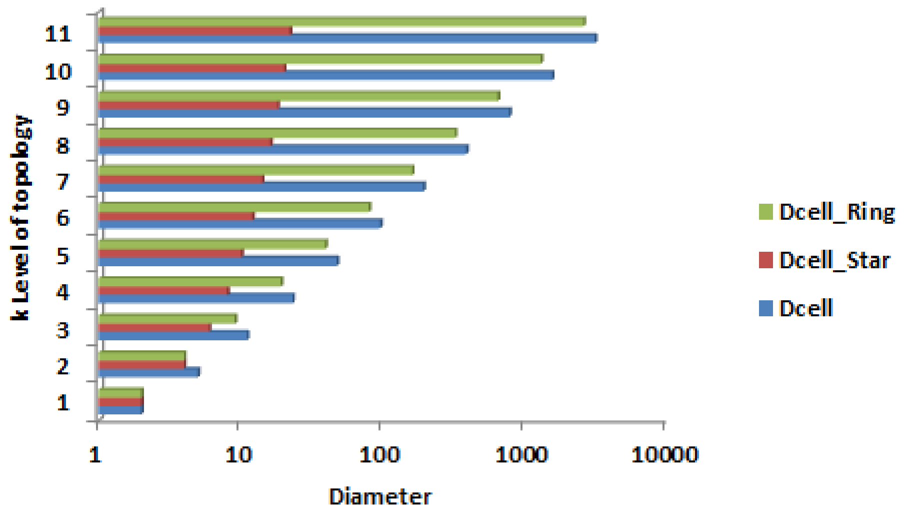


Figure 5.13: Diameter of the three topologies with $n = 4$

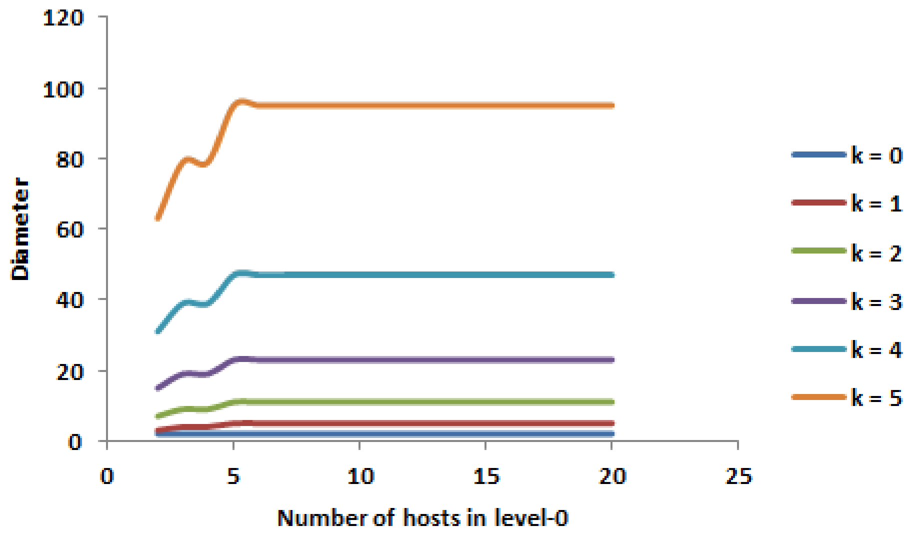


Figure 5.14: Diameter of $Dcell - Ring$ topology with varying number of servers in $Dcell - Ring_0$

Although the diameter measure in a topology indicates the longest among the shortest paths between any pair of servers, it cannot give an overview about the lengths of other shortest paths present in the topology. The metric, average shortest path length, gives better estimate of several available shortest path lengths in the topology.

In a simulation run, we assume that the switching nodes and links do not experience congestion; hence there are no discarded packets. In other words, there is no reason of dropping packets, except in the event of a failure when there is no surviving path between the source and destination host. We also assume that when one (or more) paths are available between a pair of hosts, the traffic is routed through the shortest path available between the hosts at that time. Figure 5.15 shows the performance results of $Dcell, Dcell - Star,$ and $Dcell - Ring$ topologies in the event of link failure. The link failure is applied randomly in the network with independent and equal probability for each link. The same setting of the network topology, as described in Section 4.2, is used. The simulation is conducted to estimate the average path length from one server to all other servers with link failure ratio from 0 to 0.3. As shown in Figure 5.15, the average shortest path length in $Dcell - Ring$ topology increases with the slowest ratio followed by $Dcell - Star$, for link failure ratio ranging from 0 to 0.3. The increase in the average shortest path length is from 4.05 to 4.44 in $Dcell - Ring$ and from 4.21 to 4.65 in $Dcell - Star$. However, the average shortest path length increases from 4.52 to 5.05 in $Dcell$.

Figure 5.16 shows the results of the average shortest path length as the server failure ratio increases. The server failure is also chosen randomly with independent and equal probability. The simulation results are shown for the average shortest path length against the server failure ratio ranging from 0 to 0.3 when traffic is routed from a single server to only reachable servers. The same network configuration and parameters, as mentioned previously in Section 4.2, are used. From Figure 5.16, it can be observed that the $Dcell - Ring$ topology has the smallest increase on average shortest path length taking values from 4.05 to 4.2 when the server failure ratio increases from 0 to 0.3. The average shortest path length of $Dcell - Star$ topology increases from 4.21 to 4.4; while the $Dcell$ topology has the largest increase in the average shortest path length from 4.5 to 4.8 for the server failure ratio ranging from 0 to 0.3.

The main reason behind the lower average shortest path lengths in the proposed topologies is that due to the addition of additional central switch and extra links, there are many parallel paths with the same or near-equal path lengths available in the topologies. These features provide graceful degradation in the performance in the event of link or server failures when an alternate path of relatively equal length is most probably available. Therefore, both average throughput and packet latency will not be impacted severely in the event of a failure. While a variety of paths length between two servers is available in the standard $Dcell$ topology, the average path lengths are longer as compared with the two proposed topologies. Therefore, any link or server failure in $Dcell$ will have a large effect on the increase of the path length which, in turn, will affect the overall performance. One can say that the effect of failure on the overall performance in $Dcell - Star$ and $Dcell - Ring$ topologies is less severe as compared to the standard $Dcell$.

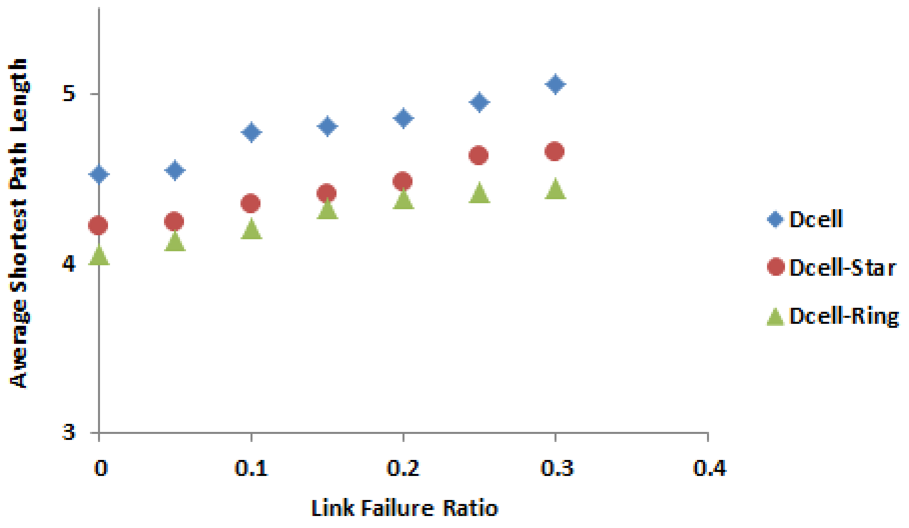


Figure 5.15: Average path length vs. link failure ratio with $n = 4$ under One-to-All traffic model

For the performance measurement of the three topologies in the event of failure as the total number of hosts increase, we set the same parameters for the topologies, as discussed in Section 4.2, except that number of hosts in a mini-switch is increased to 6 (i.e., $n = 6$). In this case, the total number of hosts increases to 42. Figure 5.17 and 5.18, respectively, shows the performance of the three topologies in the event of link and server failure. From the two figures, we find that the average shortest path length in $Dcell - Ring$ topology exceeds that in $Dcell - Star$ but remains less than $Dcell$ topology. In general, the average shortest path length of the three topologies increases with increasing the link or server failure ratio. However, the

$Dcell$ topology has the largest average shortest path length, while $Dcell - Star$ shows the smallest one over the failure ratio ranging from 0 to 0.3. These results are analyzed and compared with the results from mathematical equations ((1) to (10)), with and without link/ server failures. It is found that results from the simulation validate the one with analytical models.

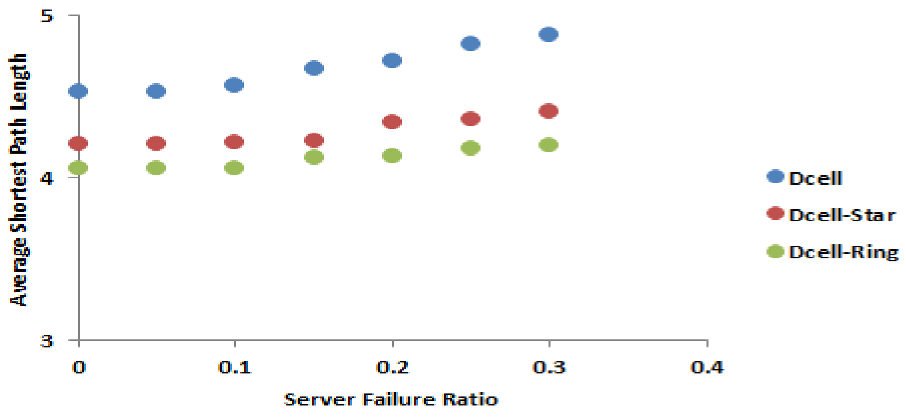


Figure 5.16: Average path length vs. server failure ratio with $n = 4$ under One-to-All traffic model

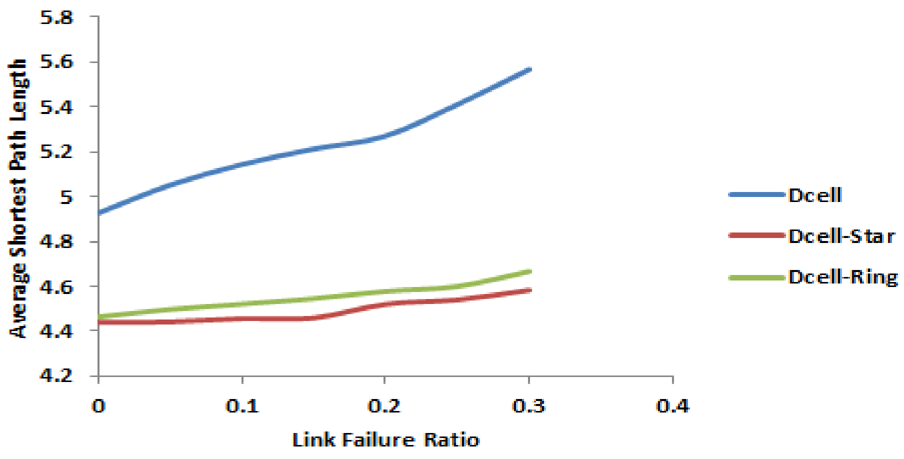


Figure 5.17: Average path length vs. link failure ratio with $n = 6$ under One-to-All traffic model

5.3. Simulation Results for the Proposed GA Path Selection Algorithm

When the optimal path(s) generated by the GA algorithm are used, the path(s) is/are deployed in the RYU controller in the Mininet simulator in order to route the packets from a certain source to a certain destination or from one source server to all other hosts.

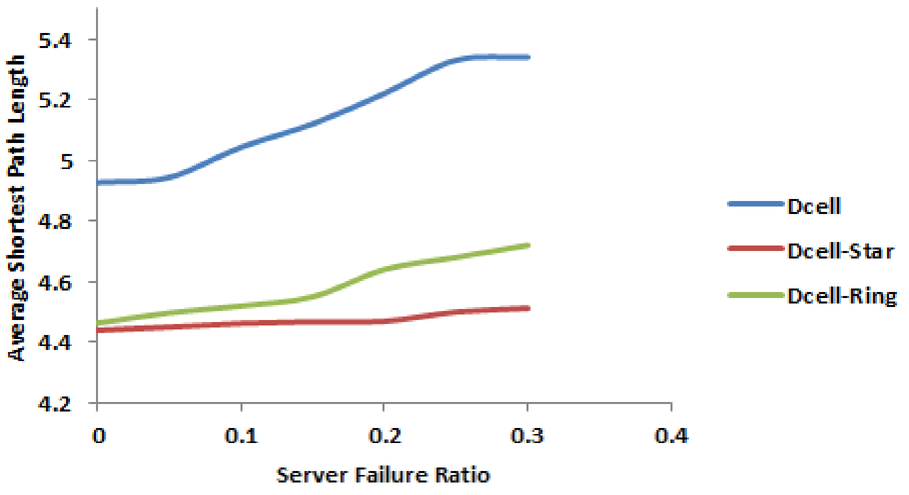


Figure 5.18: Average path length vs. server failure ratio with $n = 6$ under One-to-All traffic model

Figure 5.19 and 5.20 show the result of throughput utilization and latency, respectively, for the standard topologies under different traffic models using the optimal path obtained from GA path selection algorithm. The results show good improvement in both throughput and latency, especially for $Dcell$ topology. $Dcell$ and $Bcube$ topologies outperform Fat-Tree topology with respect to throughput in most of the traffic models. Moreover, an increase in throughput, in general, for all topologies is observed especially in One-to-All traffic model compared to the previous results from the Mininet POX controller shown in Figure 5.1. The same configuration settings are used as mentioned in Section 4.2.

As shown in Figure 5.2, $DCell$ topology has the highest latency followed by $BCube$ topology, and then the Fat-Tree topology. As shown in Figures 5.19 and 5.20, the proposed path selection algorithm using GA results in higher throughput and reduction in latency for both $Dcell$ and $Bcube$ topologies, despite the fact that the same configuration is used and a combo-switch is connected to each host.

In addition, we apply the proposed GA path selection algorithm to the two proposed topologies, $Dcell - Star$ and $Dcell - Ring$ to compute the optimal paths. Figure 5.21 shows the normaized throughgput results for different traffic models. We can conclude that both proposed topologies produce better results for all traffic patterns as compared to the standard *Dcell*. Furthermore, there is an improvement in the throughput utlization for the three topologies compared to the results shown in Figure 5.8.

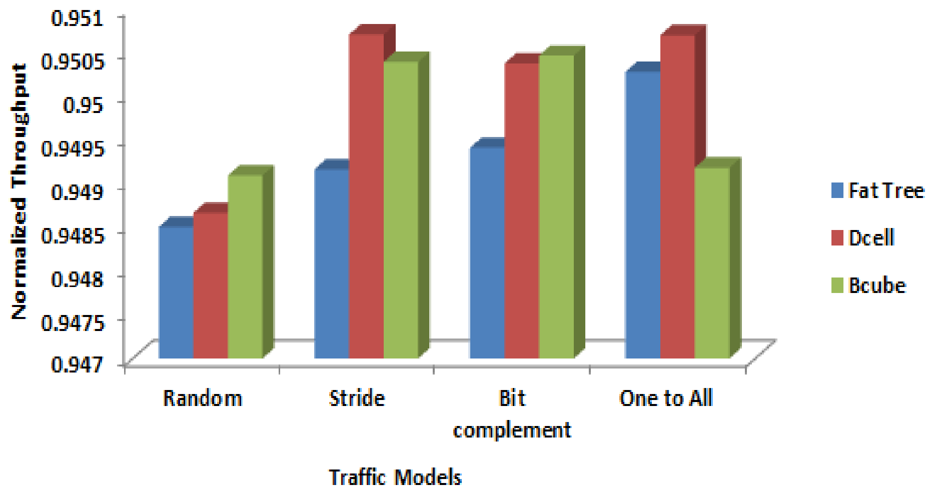


Figure 5.19: Throughput comparison for different topologies with different traffic types using the proposed GA path selection algorithm

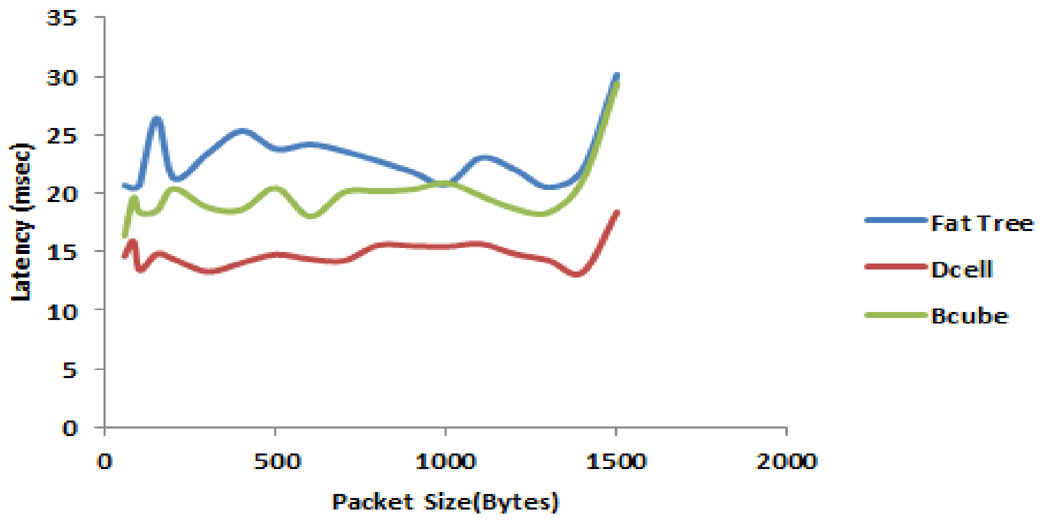


Figure 5.20: Latency comparison for the topologies with different packet size under All-to-All traffic model using the proposed GA path selection algorithm

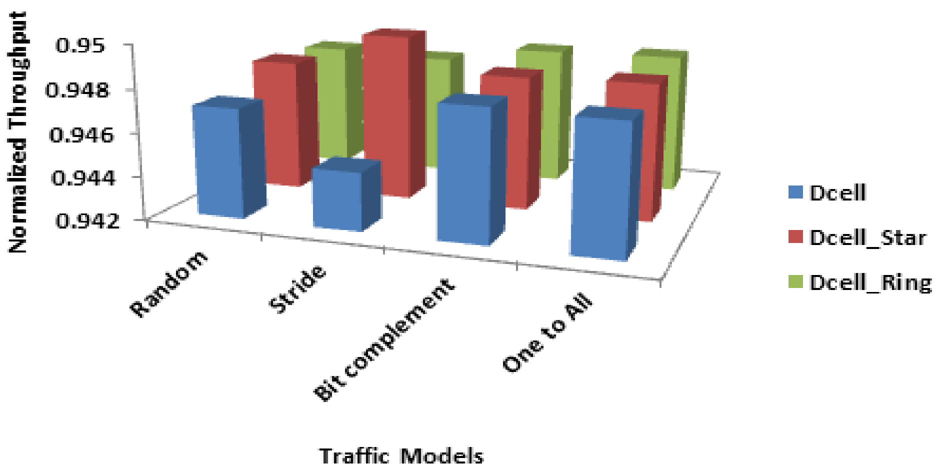


Figure 5.21: Throughput comparison for standard $Dcell$ and the two proposed topologies with various traffic types using the proposed GA path selection algorithm with $n = 4$

5.4. Comparison and Discussion

In this subsection, we compare the performance results from the two proposed architectures with well-known, standard, network topologies. Additionally, we present how the use of GA algorithm in selecting the optimal path affects the performance of DCN topologies.

By comparing different DCN topologies using the simulation tool (Mininet), we conclude that $Dcell$ topology performs the worst in terms of throughput and latency metrics. However, the main advantages of using $Dcell$ topology over the others are better scalability and fault-tolerance. We can improve the performance of standard $Dcell$ architecture with respect to fault-tolerance by either modifying the architecture to support multiple shortest paths between any two servers, and/or implement a better fault-tolerant path selection algorithm for choosing the best path for routing the packets. For the two proposed topologies, $Dcell - Star$ and $Dcell - Ring$, the simulation results show an improvement in the throughput utilization compared with the standard $DCell$, and the percentage of improvement increases as we increase n. The proposed topologies show an improvement in the throughput in the range of 3% to 5% (depending upon traffic model) for $n = 16$ with respect to the standard $DCell$. This is mainly due to provisioning and utilization of multiple alternate paths between any pair of servers. In addition, the proposed topologies provide a graceful degradation in the performance in the presence of link or server failures.

For further reduction in latency and increase in the throughput, we can use GA algorithm as an optimization mechanism for selecting the optimal path with minimum number of hop counts between two hosts in One-to-One and One-to-All traffic models. We have observed an enhancement from 1% to 2% in the throughput compared to that obtained with POX controller when the GA algorithm is applied to find the optimal path between hosts for $n = 6$, depending upon the traffic model used. We can expect higher percentage of enhancement when n is increased. Tables 5.1 and 5.2 compare the throughputs in the topologies when RYU controller is used instead of the POX controller for the two traffic models (Stride and One-to-All) for $n = 6$.

Table 5.1: Improvement in the throughput using RYU controller over the POX controller for the Stride traffic model with $n = 6$

Topology	Normalized Throughput from RYU Controller	Normalized Throughput from POX Controller	% percentage of Improvement
Dcell	0.943	0.927	~2%
Dcell – Star	0.944	0.934	~1.1%
Dcell – Ring	0.9436	0.932	~1.3%

Table 5.2: Improvement in the throughput using RYU controller over the POX controller for One-to-All traffic model with $n = 6$

Topology	Normalized Throughput from RYU Controller	Normalized Throughput from POX Controller	% percentage of Improvement
Dcell	0.945	0.929	~2%
Dcell – Star	0.95	0.934	~2%
Dcell – Ring	0.948	0.933	~2%

Chapter 6. Conclusion and Future Work

Due to the tremendous increase in the use of cloud-based services, the volume of data traffic to the data centers is on constant rise. Data center network (DCN) topologies provide a network within a datacenter connecting a huge number (~10,000) of servers with several switches and other networking devices. The most important features that a network topology must have are scalability, fault-tolerance, and QoS support for the desired applications. The provision of fault-tolerance is becoming even more critical as the number of servers and devices increase so does the probability of failures. Therefore, there is a strong need to have the support for fault-tolerance in the DCN architecture.

In this thesis, we propose two new DCN topologies, called $Dcell-Star$ and $Dcell-Ring$. The proposed topologies are derived from the standard $Dcell$ topology with slight modifications in the original architecture. The modifications add a set of alternate shortest paths between any two servers in the topologies. The proposed topologies provide higher throughputs and lesser latencies as compared to the standard $Dcell$ topology. The improvement in the throughput ranges from 2.5% to 5% (depending upon the traffic model used) for $n = 16$. For large values of n, one can expect higher percentage of improvements in throughput. Furthermore, as compared to the standard $Dcell,$ the $Dcell-Star$ topology has lesser diameter and lesser average shortest path length which makes it superior for using it in a large network with a large number of servers in $level-0$. $Dcell-Ring$ topology can be used for small-scale network, especially when $n \leq 5$. Reduction in the diameter metrics, especially in $Dcell-Star,$ results in low-latency for routing packets, even with large number of servers. In addition, the proposed topologies offer graceful performance degradation in case of a link or server failure. We also present a new mechanism using GA algorithm for selecting the optimal paths with minimum number of hop counts in the two traffic models, One-to-One and One-to-All. When GA is applied to the standard topologies, it is found that the performance of $DCell$ topology improves in terms of throughput and latency. Finally, we compare the results after applying the GA path selection algorithm to the proposed topologies with the results produced by the Mininet simulations. It is concluded that there is an improvement of about 2% in throughput utilization when GA algorithm is applied for selecting the optimal routes.

Since energy consumption in datacenters is a serious problem, the work presented in this thesis can be extended to investigate new DCN energy-efficient topologies. Moreover, investigating efficient algorithms to migrate Virtual Machines in the proposed topologies in order to save energy could be researched.

References

[1] M. Al-Fares, A. Loukissas and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, p. 63, 2008.

[2] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya and A. Vahdat, "PortLand: A Scalable Fault-tolerant Layer2 Data Center Network Fabric," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, p. 39, 2009.

[3] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang and S. Lu, "Dcell: A Scalable and Fault-tolerant Network Structure for Data Centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, p. 75, 2008.

[4] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang and S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, p. 63, 2009.

[5] G. Wang, D. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch and M. Ryan, "C-Through: Part-time Optics in Data Centers," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, p. 327, 2010.

[6] Y. Liu, J. Muppala, M. Veeraraghavan, D. Lin and M. Hamdi, *Data Center Networks: Topologies, Architectures and Fault-tolerance Characteristics*, 1st ed. Cham: Springer International Publishing, 2013.

[7] C. Leiserson, "Fat-trees: Universal Networks for Hardware-efficient Supercomputing," *IEEE Transactions on Computers*, vol. -34, no. 10, pp. 892-901, 1985.

[8] R. Harris, "Fat trees and skinny switches," Storagemojo.com, 2008. [Online]. Available: https://storagemojo.com/2008/08/24/fat-trees-and-skinny-switches. [Accessed: 12- Nov- 2016].

[9] R. Chirgwin, "Which data centre network topology's best? Depends on what you want to break," Theregister.co.uk, 2015. [Online]. Available: http://www.theregister.co.uk/2015/10/13/which_data_centre_topology_is_best_depends_on_what_you_want_to_break/. [Accessed: 12- Nov- 2016].

[10] A. Andreyev, "Introducing data center fabric, the next-generation Facebook data center network," Facebook Code, 2014. [Online]. Available: https://code.facebook.com/posts/360346274145943/. [Accessed: 12- Nov- 2016].

[11] Y. Sun, J. Chen, Q. Liu and W. Fang, "Diamond: An Improved Fat-tree Architecture for Large-scale Data Centers," *Journal of Communications*, vol. 9, no. 1, pp. 91-98, 2014.

[12] Z. Li, Z. Guo and Y. Yang, "BCCC: An Expandable Network for Data Centers," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3740-3755, 2016.

[13] "Survival of the fittest", En.wikipedia.org, 2014. [Online]. Available: https://en.wikipedia.org/wiki/Survival_of_the_fittest. [Accessed: 12- Nov- 2016].

[14] N. Mansour and K. El-Fakih, "Simulated Annealing and Genetic Algorithms for Optimal Regression Testing," *Journal of Software Maintenance: Research and Practice*, vol. 11, no. 1, pp. 19-34, 1999.

[15] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang and M. Zhani, "Data Center Network Virtualization: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909-928, 2013.

[16] L.Brian, M.Aman, and T.Niharika. "A Survey and Evaluation of Data Center Network Topologies," Available: https://arxiv.org/abs/1605.01701, 2016.

[17] K. Bilal, S. U. Khan, J. Kolodziej, L. Zhang, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang and D. Chen, "A Comparative Study of Data Center Network Architectures," *European Conference on Modeling and Simulation (ECMS)*, pp. 526-532, 2012.

[18] Y. Liu, D. Lin, J. Muppala and M. Hamdi, "A Study of Fault-tolerance Characteristics of Data center networks," *Dependable Systems and Networks Workshops (DSN-W)*, 2012 IEEE/IFIP 42nd International Conference on. IEEE , pp. 1-6, 2012.

[19] Y. Liu and J. Muppala, "Fault-tolerance Characteristics of Data Center Network Topologies Using Fault Regions," *Dependable Systems and Networks (DSN)*, 2013 43rd Annual IEEE/IFIP International Conference on, Budapest, Hungary, 2013.

[20] F. Yao, J. Wu, G. Venkataramani and S. Subramaniam, "A Comparative Analysis of Data Center Network Architectures," *Communications (ICC)*, 2014 IEEE International Conference on. IEEE, 2014.

[21] M. Team, "Mininet: An Instant Virtual Network on your Laptop (or other PC) – Mininet," Mininet.org, 2015. [Online]. Available: http://mininet.org/. [Accessed: 12- Nov- 2016].

[22] T. Chen, X. Gao and G. Chen, "The features, Hardware, and Architectures of Data Center Networks: A survey," *Journal of Parallel and Distributed Computing*, vol 96, pp. 45–74, 2016.

[23] K. Miettinen, *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming, and Industrial Applications*, 1st ed. Chichester: Wiley, 1999.

[24] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*. New York: Wiley, 2000.

[25] S. C.Abraham and G. Dutt Shukla, "Shortest Path Computation in Large Graphs using Bidirectional Strategy and Genetic Algorithms," *International Journal of Computer Applications*, vol. 109, no. 13, pp. 27-30, 2015.

[26] A. Hamed, "A Genetic Algorithm for Finding the k Shortest Paths in A Network," *Egyptian Informatics Journal*, vol. 11, no. 2, pp. 75-79, 2010.

[27] A. Chaudhary and N. Kumar, "Genetic algorithm for Shortest Path in Packet Switching Networks," *Journal of Theoretical and Applied Information Technology*, vol 29, no. 2, pp. 107-117, 2011.

[28] L. Ferraz, D. Mattos and O. Duarte, "A Two-phase Multipathing Scheme Based on Genetic Algorithm for Data Center Networking," *Global Communications Conference (GLOBECOM)*, 2014 IEEE. IEEE, 2014.

[29] J. Touch and R. Perlman, "Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement," no. 5556. IETF, 2009 [Online]. Available: http://www.ietf.org/

[30] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," no. 2992. IETF, 2000 [Online]. Available: http://www.ietf.org/

[31] Kim, M. Caesar and J. Rexford, "Floodless in Seattle: A Scalable Ethernet Architecture for Large Enterprises," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, p. 3, 2008.

[32] J. Khalid and M. Mohammed, "Selection Methods for Genetic Algorithms," *International Journal of Emerging Sciences*, vol 3.4, pp. 333-344, 2013.

[33] "POX Controller Tutorial | SDN Hub," Sdnhub.org, 2011. [Online]. Available: http://sdnhub.org/tutorials/pox/. [Accessed: 22- Nov- 2016].

[34] "OpenFlow Tutorial - OpenFlow Wiki," Archive.openflow.org, 2011. [Online].Available:http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial. [Accessed: 12- Nov- 2016].

[35] "Router — Ryubook 1.0 documentation," Osrg.github.io, 2014. [Online]. Available: https://osrg.github.io/ryu-book/en/html/rest_router.html#setting-static-routes-and-the-default-route. [Accessed: 25- Apr- 2017].

[36] V. GUEANT, "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool," Iperf.fr, 2013. [Online]. Available: https://iperf.fr/. [Accessed: 12- Nov- 2016].

[37] T. Fisher, "How to Use the Ping Command in Windows," Lifewire, 2016. [Online]. Available: https://www.lifewire.com/ping-command-2618099. [Accessed: 12- Nov- 2016].

[38] J. Pandey and R. Subramanian, "DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers," Reproducing Network Research, 2012. [Online].Available:https://reproducingnetworkresearch.wordpress.com/2012/06/04/dcell-fault-tolerant-routing/. [Accessed: 12- Nov- 2016].

[39] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," *NSDI*, vol. 10, pp. 19-19, 2010.

Vita

Heba Helal was born in 1989, in Jizan, Saudi Arabia. She received her secondary education in Mansoura, Egypt. She received her B.Sc. degree in Computer Engineering from the Mansoura University in 2011. From 2011 to 2015, she worked as a demonstrator at Mansoura University.

In September 2015, she joined the Computer Engineering master's program at the American University of Sharjah as a graduate teaching assistant. She was involved in lab supervision and conducting research in the area of data center network topologies. During her master's study, she co-authored three papers which were presented in international conferences. Her research interests are in Computer Networks, Software Engineering and IoTs. Her work on datacentre topologies has been presented in the Seventh International Conference on Modeling, Simulation and Applied Optimization (ICMSAO ‘17) held in April 2017 in Sharjah, UAE.