Sustainable and Resilient Network Infrastructure Design for Cloud Data Centers

Ritchie Qi

A thesis submitted to

Auckland University of Technology

in partial fulfilment of the requirements for the degree

of

Master of Computer and Information Sciences (MCIS)

2016

School of Engineering, Computer and Mathematical Sciences

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”

Signature of Candidate: ……

Acknowledgements

I would like to express my deep gratitude and thanks to my supervisor, Dr William Liu, of the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology. He has a strong sense of responsibility and dedication not only for my thesis supporting, but supervising me of great sense of logical thinking, which is great fortune to me. In every problem I came across, he showed me very easy-to-follow advices and examples which give me the greatest help.

I am also deeply grateful to my co-supervisor, Associate Professor Jairo Gutierrez, he can always give me the correct direction to indicate me where I should go. His very effective recommendations always inspire me on my thesis forming and development. In each NSRG meeting, I always paid my attention to listen to his every comment. His logical and “targeted” thinking way is very helpful for me to follow.

I should like to express my deepest gratitude to Dr. Asad Waqar Malik whose kindness and advice have made this work possible. He gave me valuable instructions on his simulator: CloudNetSim++. His effective advice, shrewd comments have kept the thesis in the right direction.

I owe my loving thanks to my family. Without their encouragement and understanding it would have been impossible for me to finish this work.

A sincere thank is given to Network and Security Research Group (NSRG) for their valuable suggestions and feedback to my research and conference. Without their help, I would not think of that I can publish my paper on INFOCOM 2016 Workshops.

Publications

Ritchie Qi, William Liu, Jairo Gutierrez and Asad Malik, Crash Me If You Can: Rethinking Sustainable Data Center Networking From a Topological Perspective, 2016 Green and Sustainable Networking and Computing Workshop in IEEE INFOCOM16, San Francisco, 9-15 April, paper accepted and being invited for presentation.

Abstract

A datacenter is a pool of resources such as computational, storage, and servers interconnected using a communications network. Data Center Networking (DCN) holds a pivotal role, and it needs to be scalable and efficient to connect the growing number of servers so as to handle the intensive demands of cloud computing. Recently there has been a rapidly growing field of literature on DCNs, but it mainly focus on studying how to model and evaluate the resource provisioning and allocation algorithms for more effective and efficient resource management of a cloud system. Unfortunately, there are not many studies that reveal how the underlying network`s topological connectivity can affect the DCNs` performance, in areas such as energy consumption and service resilience. There is a saying that 'it is not what you know but who you know' i.e., algorithm, connectivity, which argues that people get ahead in life based on their connections, not on their skills or knowledge, and every day offers evidence of this proverb. This case also applies to DCNs. DCNs performance is not merely a function of resource provisioning and allocation, but also it is a network-wide activity. The structure and ties that link a data center to other data centers are also critical factors.

In this thesis, the researcher has proposed a method for evaluating topological metrics (network robustness metrics and node centrality metrics) to identify critical nodes and edges in a network so that it measures the overall DCN network performance change (throughput, latency, packet drop ratio) according to the faults on the network. We have identified the energy changes according to the change of internal DCNs; the simulation study showed that the traffic load has a large impact on energy consumption. Apart from that, the state of the art for modern DCNs is elaborated that depicts the true picture of the current progress in this field where good research can actively contribute.

Chapter 1 Introduction

The rise in information service provisioning is a product of user demands, and the rapid growth of users` demands in obtaining data has pushed Information Communications Technologies (ICT) services to a whole new level. This has led to a new evolution of cloud computing services. The prominent characteristic for a cloud is its centralized distribution which applies high performance computing to the task it receives, and delivers services by sharing resources which are stored inside a cloud rather than having local servers to handle applications traditionally [1, 2]. This “pay-as-you-go” model has brought huge benefits to customers, but there is an enormous cost associated with maintaining its infrastructure. The cost has to be considered seriously, as this leads to a threshold level for those organizations intending to provide such services. Currently, the well-known cloud service providers are Amazon, Microsoft and Google [2].

At present, cloud services are still evolving and, as a consequence, are far from perfect. Although the cloud services are still running with the aid of network infrastructure, cloud performance varies and is still being impacted on by different layers. Numerous cloud competitors have made huge efforts to alleviate the limitations of clouds, such as working on bandwidth and internal management. Data center Networking (DCN) is a vital component of the cloud infrastructure and it is being increasingly adopted by organizations to handle the core business and operational data that interconnects all the components in the cloud, while delivering main cloud services such as data storage and protection. It follows that the maintenance work turns out to be extremely significant to cloud service providers. Therefore, DCN performance is the top priority for most cloud providers.

In addition, a robust network topology is essential to comprehensively address malicious attacks or nodes and link failures. Anecdotally, many cloud hosting providers have stated that they have experienced network failures to varying degrees [3]. One company running 100-200 nodes hosts on a major cloud hosting provider

declared that in a three-month period, the provider's network came across five distinct periods of partitions which made some connectivity unavailable between the provider's cloud network and the public Internet, and others isolated the cloud network from the provider's internal managed-hosting network. According to a report from Amazon, a total partition between the front-end and back-end servers [4] caused a severe network disruption in the Amazon cloud network, and the site's servers lost their connections to all back-end nodes for a few seconds, but several times a month. Despite the disruptions being short, they resulted in 30-45 minutes outages and a corrupted index for ElasticSearch. As problems escalated, the outages occurred two to four times a day. Furthermore, in December 2012, a regular software update on an aggregation switch caused instability at Github [5]. The engineers consequently killed that particular software agent running on one of the aggregation switches. However, this movement stopped other aggregation switches from handling link aggregation and spanning tree, resulting in all traffic between access switches being blocked for 90 seconds. The 90-second stoppage caused failover chaos that led to parts of files becoming unavailable, and delays in delivering messages among file servers. Recovering those downed file-server pairs took five hours, during which Github`s service was critically compromised.

Network failures cannot be eliminated due to the complicated network system, while the failure of failover policy can cause unpredicted consequences. Therefore, a high-standard failover plan is necessary to mitigate the probability of failure occurring. However, except for targeted systematic recovery, a comprehensive analysis of network performance change is also required as a metric in risk assessment and is a necessary precaution. Moreover, the energy consumption metric is also a critical concern that needs to be addressed as it can be a measurement used to predict expenditure of the entire network.

1.1 Background

Cloud computing is a model for enabling convenient, ubiquitous, on-demand network access service to a shared pool of configurable computing resources (e.g., networks, storage, servers, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. Cloud hosts a variety of applications from consumer, scientific and business domains ranging from computing intensive data storage to intensive applications. Data centers are infrastructure that hosts cloud applications. The trend of growing reliance on online services, and the increasing demand in mobile device has already converted many client-based applications into cloud services [6]. To guarantee the service level, cloud service providers have to take several factors into consideration, delivering cloud service is dependent on the network infrastructure, while the services are always delivering in a geo-distributed network. Improving such big area networking is not an easy job, so that the main emphasis of the work should be elaborated on what they can control. Therefore, data centers as the service-originated is paid extra efforts to maintain the service provisioning. Server farm placed inside data centers should be interconnected in a very efficient way to improve the service provisioning performance. Also, the efficient server arrangement and the appropriate placement can largely reduce the power consumption and the heat dissipation [7].

Moreover, with the sustained growth in computing capacity, the cost and operational expenses (OPEX) are showing a sharp increase. Energy consumption has been a great concern for data centers' operators. According to the survey conducted by Gartner Group [8], approximately 40% of data center OPEX comes from the energy consumed by Information and Communications Technology (ICT) equipment, which is composed of computing servers (2/3) and communication links (1/3). The remainder 60% energy consumption comes from cooling and power distribution, which are 45% and 15% respectively. On the other hand, the cooling cost of heat

generated by data center infrastructure ranges between $2 to $5 million per year [9, 10]. Therefore the optimized data center architecture plays an important role in OPEX reduction [11].

1.2 Motivation

Datacenters are very energy hungry and consume huge amounts of electricity, resulting in high operational cost and big carbon foot print to the environment. Therefore, Green Cloud solutions are in need that efficiently manages energy consumption to reduce operational costs without violating negotiated Quality of Service (QoS) and Service Level Agreements (SLAs) in the cloud computing environment. In cloud computing, it is a bit complex task to efficiently allocate different ICT resources e.g. CPU, servers, storage disks) due to presence of number of heterogeneous nature applications e.g. web apps, Content Distribution Network (CDN) etc. that have contentious resource allocation requirements. So far a good effort has been put to better address the problem of efficiently allocating resources in cloud with different level of success. Studies have shown that an idle server consumes almost 70% energy to the one that is working at full speed. So far the main focus was to improve performance and a lot of efforts are done in this area, but now energy efficiency at datacenters needs a serious attention as well. As those datacenters not only consume a high amount of energy but also emit CO_2 that is also acting as a critical role in affecting environment. Apart from network performance, energy efficiency is also needed to be considered. If we successfully deliver some good techniques, it will help service providers not only save operational cost but will also play a positive role towards environment safety. It clearly justifies that it not only contributes positively but also help other problems fixed. The main idea is to turn the idle server switched off or to sleep mode, with the technique of adjusting CPU voltage/frequency. By this way a good amount of energy can be saved but SLAs need to meet at the same time.

While the current studies on DCNs are mainly focusing on how to develop more effective and efficient resource provision and allocation algorithms among data

centers, but there is not much discussion on how the underlying topology change can affect the overlay DCNs` performance.

Except the various DCs` network performance evaluation, another main purpose of this thesis is to identify the energy changes according to the change of internal DCN. The energy-saving techniques then are analysed in a detailed way. Apart from that, the state of art for trendy DCNs is elaborated that depicts the true picture of the current progress in this field where a good research can actively contribute.

1.3 Research Questions

This thesis is conducted surrounded by three main doubts which are shown as follows, Chapter 3 and 4 uses both modelling and simulation techniques to elaborate the process of conducting researches, the answers are then summarized in the conclusions.

Q 1: Does the underlying DCN topology have an impact on the cloud network performance (QoS & energy)?

Q 2: What network topological metrics can be used as an index to quantify the cloud data center performance in the cases of energy efficiency and QoS?

Q 3: How to use this metric to determine the critical nodes/links in a data center network?

1.4 Contributions

In current research, there have not been enough contributions on the concrete comparison of various DCNs based on the evaluation of data center network performance, such as the network throughput, dropped packet, network latency, nor energy consumption estimation. Current research only focuses on either the DCN network-aware performance or the energy-aware. For example, in [12], the authors evaluate the reliability of popular DCNs (Fat-tree, BCube-3&5 switch ports, DCell-3&5 switch ports) by using different network reliability metrics according to the Link Failure Rate (LFR) from a topological view. In [13], DCN energy efficiency

is evaluated by analyzing resource allocation techniques in a cloud computing system. Also, a comprehensive analysis of green solutions on the basis of current challenges that DCNs face is conducted in [14].

This thesis is focused on addressing such a shortage in the literature by combining the network robustness metrics with network performance and the currently popular topic of energy consumption, and proposes a method of evaluating node centrality metrics to find critical nodes and edges so that one can measure a DCN network`s performance changes according to the faults on the network`s critical paths.

Firstly, a topological comparison has been made between several popular DCN structures, including Fat-tree, Three-tier, BCubes (different layers), and HyperFlatNet. This thesis compares BCube - 2 layers with 3 layers; and important differences occurred between switch port number and switch number. BCube – 2 layers uses far fewer switch numbers than 3 layers, which significantly helps the network service provider reduce the expenditure cost. Moreover, Fat-tree and Three-tier have two similar DCN structures, and this thesis gives a comprehensive picture of both DCN topologies. HyperFlatNet is a brand new DCN model in this area; this thesis also looks at its network performance and energy consumption in comparison with other types of DCN topologies.

Secondly, a method of evaluating node centrality metrics is proposed to find critical nodes and edges so that measures of the DCN network`s performance changes according to the faults on the network`s critical paths. Network fault tolerance is a metric of measuring the robustness of a network; different DCN architectures have various performance responses to network failure. However, such responses can be measured or even predicted by determining robustness and node centrality metrics. For example, if the first aggregation switch (AS[0]) is determined as the most critical node, then the failure of this node can cause the largest network performance change.

1.5 Thesis Structure

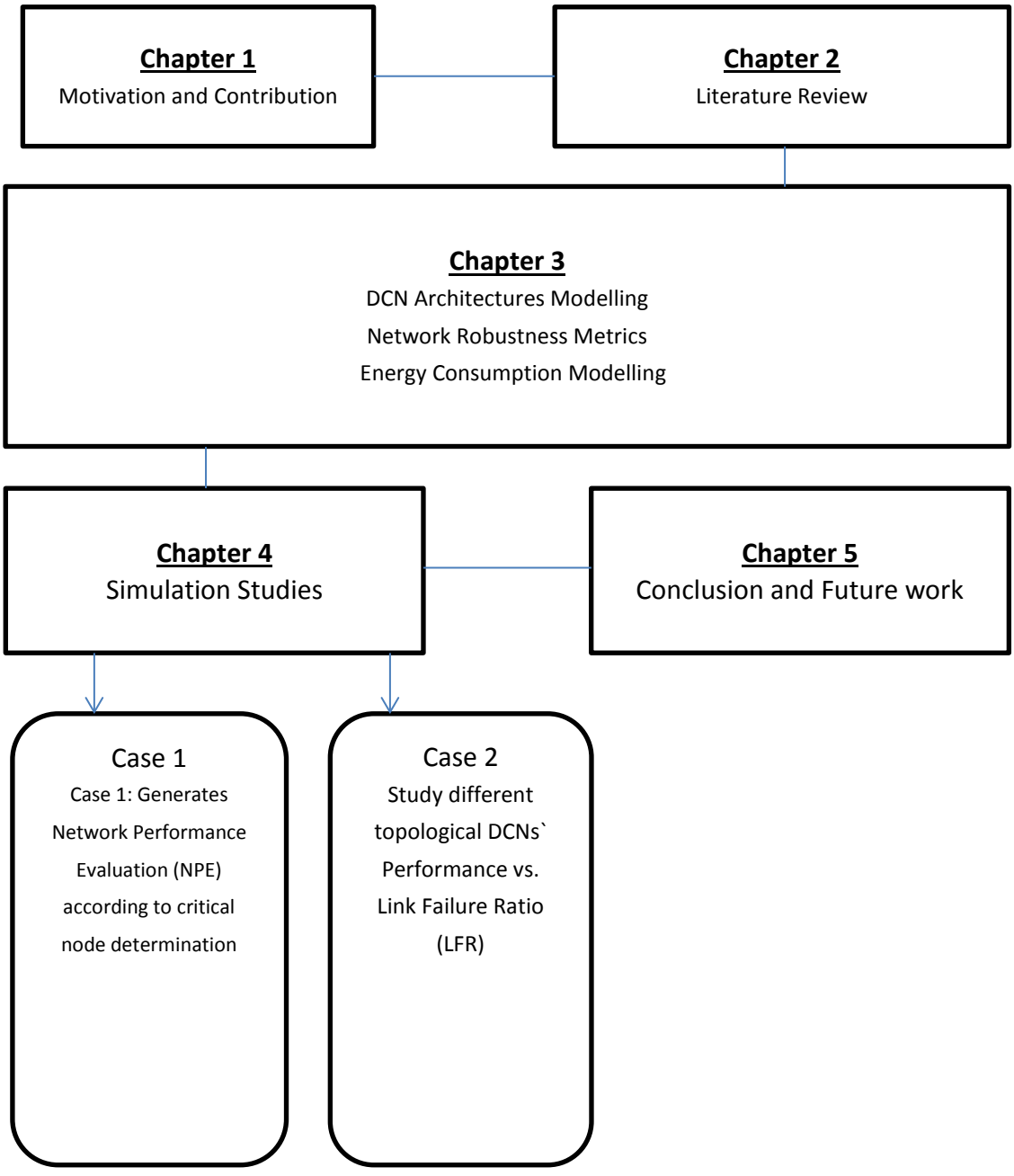


Figure 1.1 – Thesis structure

The thesis structure is depicted in Figure 1.1. The reasons for conducting this research are described in the introduction, and the background of current

problems is illustrated in the second section, followed by the motivation for doing this thesis. The contributions are summed up in the consequent section.

In **Chapter 2**, the background of cloud computing and data center networks is introduced. Five topological types of data center networks are analyzed for their interconnection from a topological aspect. The characteristics for each type of data center network are elaborated, which is useful for the data center architectures modeling in Chapter 3. Furthermore, energy saving techniques adopted in DCNs, including dynamic voltage/frequency scaling (DVFS) and dynamic power management (DPM), are presented so that the algorithms for both techniques are established in the energy modeling section. In addition, the data center traffic characteristics and estimation to set up parameters for simulation studies are detailed in Chapter 4.

In **Chapter 3**, each data center topological structure is modelled according to its characteristics; for example, a 64-server DCN is adopted to enable comparisons among five DCN structures. Then, the network graph modelling is presented and formulated for network robustness metrics measurement and analysis.

In **Chapter 4**, there is a discussion around the two case studies that have been conducted. Within each case study, several scenarios are implemented for more extensive comparison studies. The first case studied the correlations between the topological location of nodes and overall network performance and energy consumption. While in the second case study, more realistic (random) data center traffic is simulated on various DCN topologies, i.e., Fat-tree, Three-tier, BCube-2 layer, BCube-3 layers, and HyperFlatNet. The simulation was conducted by increasing the failure ratio so that each DCN`s capacity for fault tolerance was revealed.

Chapter 5 summarizes the final findings and also lists the constraints of the current work. Then possibilities for future research are discussed.

Chapter 2 Related Work

2.1 Background

Cloud computing is a natural evolution of the widespread adoption of virtualization, service-oriented architecture (SOA), and computing resources which are based on networking [15]. Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [16]. The National Institute of Standard and Technology (NIST) [17] states that the cloud computing resources are located in the data center and those end users can reach to data centers through services and infrastructure providers. Therefore, there are three components in the cloud system: end users, services and infrastructure providers, and data centers. These three parts are connected together by network infrastructure. Moreover, there are three types of service models in cloud computing. These are “Software as a Service (SaaS)”, “Platform as a Service (PaaS)” and “Infrastructure as a Service (IaaS)”. The SaaS is a provider that supplies remotely run software packages; it is a pricing model that is offered via the internet to consumers. The PaaS is a provider that offers an additional layer on top of virtualized infrastructure. This software platform can be deployed in exchange for built-in scalability. The IaaS is a provider that allows physical resources to be assigned and split dynamically by provisioning capacity in virtualization [18].

There are an increasing number of companies that have already transferred their business to a cloud platform so as to mitigate the burden of management and maintenance of different resources [19–21] and to allow the supplementation of their assets. Many industry flagships such as Microsoft, Amazon, Gogrid, vCloud Express, Layered Technologies, ENKI Prima Cloud and Flexiant offer the service of resource integration, platform provisioning, and infrastructure outsourcing.

Furthermore, cloud computing follows a business model of “pay-as-you-go” strategy; the cloud users only pay for the services as they use them and based on the service type [22]. In a cloud environment, the ICT resource capacity can be increased or removed via the invocation of a Simple Object Access Protocol (SOAP)/Restful API. Cloud computing distributes workloads over servers and offers various services such as data access, computation, backup, and software and hardware services to end-users. The cloud providers guarantee the quality of service to the customers on the basis of service level agreements (SLAs) that charge for usage and reservation of data center resources. On the other hand, the cloud computing infrastructure has critical key issues such as ensuring security and privacy of the hosted ICT resources and application data [23], meeting performance demands despite uncertainties, dynamic reliability, standardization, fault-tolerance, debugging, scalability, reducing operational costs, and carbon emission [24].

Reducing carbon emissions for cloud computing data centers has become a dominant research topic in both academia and industry. This fact shows that the energy supplementation to datacenters for their power supply, cooling, operation, and illumination, has been increasing, which contributes dramatically to the total operational costs [25]. Reducing power consumption and energy dissipation have become significant concerns for making cloud services environmentally developable and sustainable [26]. According to the McKinsey report in 2008 [27], the amount of electricity usage in global data centers was 1.3 % of total electricity usage worldwide. The total estimated electricity cost for data centers in 2010 was $11.5 billion. Energy costs in a typical data center doubles every five years.

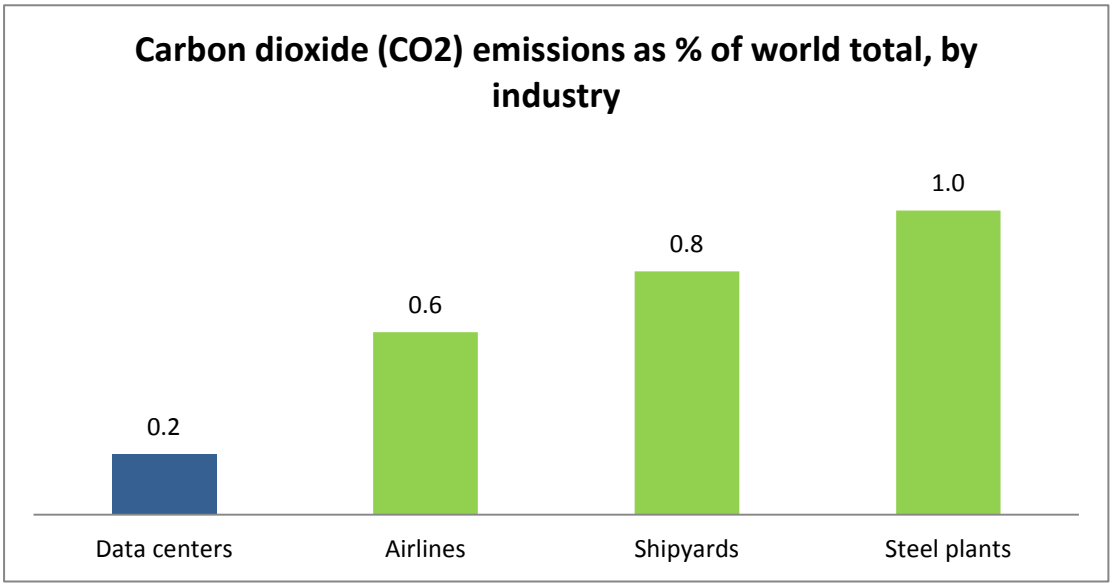


Figure 2.1 – Carbon dioxide emissions as % of world total

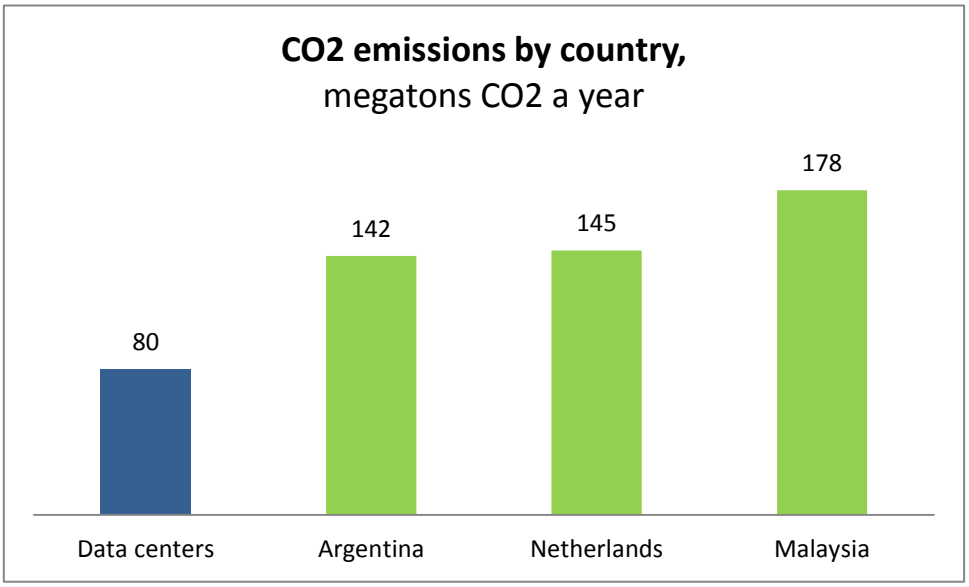


Figure 2.2 - Carbon dioxide emissions by country

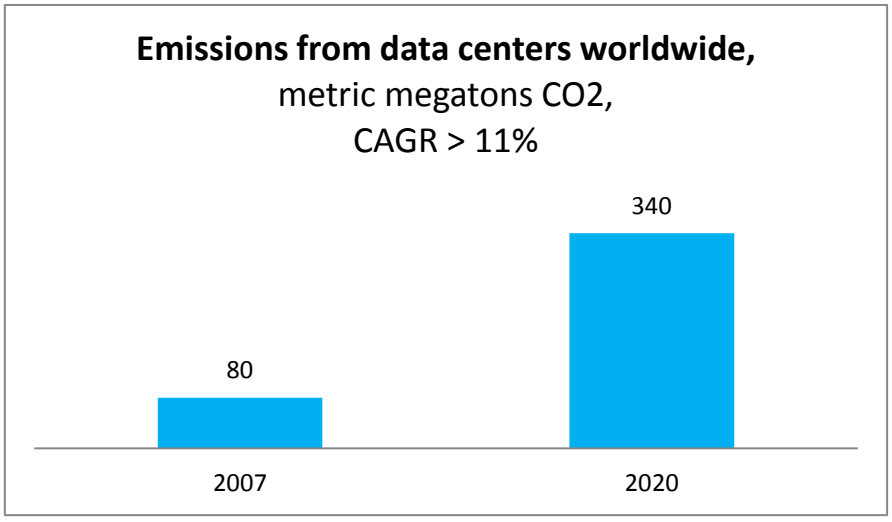


Figure 2.3 - Emissions from data centers worldwide

Figure 2.1 and 2.2 show the Carbon dioxide emissions worldwide (%) and by country, Figure 2.3 shows the carbon emission from all datacenter worldwide estimated by Stanford University, McKinsey study and Gartner research (CAGR: Compound Annual Growth Rate)

The energy inefficiency in data centers is mostly due to servers using power while idle. For example, a server still consumes over 50% of the peak load power, even at an extremely low working load of 10% CPU utilization [28]. The cloud service providers urgently need powerful energy efficient management of ICT resources in data centers due to several reasons [29]. First, the increasing electricity costs for supplying ICT resources and cooling systems has exceeded the purchase for ICT hardware. Second, increased energy usage and inefficient heat dissipation systems have a great influence on the system reliability and scalability of data center hardware. Finally, it is an environmental issue, as mentioned above, as governments are now seeking to regulate data center power usage.

As researched, there are four well known approaches for designing energy efficient cloud computing datacenters:

(a) Infrastructure of high-intensive maintenance to lower the need for equipment replacement e.g., avoid server breakdown by maintaining safe operation temperature.
(b) Increase equipment utilization (reduce the time servers are idle).
(c) Flexibly allocate resources in an infrastructure to reduce the energy dissipated.
(d) Minimize self-management and flexibility as the cost is spread across a number of datacenters [30].

To elaborate, some recent research has investigated the optimization of energy utilization by monitoring the performance of virtualized ICT resources (servers) and hosted workload under variable CPU frequency [31, 32]. Other approaches have focused on techniques of voltage adjustments by switching off unnecessary resources e.g., a display monitor, processors speed control, and using hibernate or sleep mode [33–35]. However, the energy saved by scaling down the CPU voltage is far less than

powering off a physical server. Cloud computing is a prototype shift from the outdated uniprocessor computation approach of development to that of an accessible, multi-tenant, and global infrastructure.

2.1.1 Cloud Computing Network Architecture

The cloud computing network architecture occupies a number of various elements in different layers. The cloud network architecture is established based on the data center, which is the cloud infrastructure that accounts for the cloud service provisioning. The data center core layer [36] is responsible of maintaining the connections to the remaining elements in a data center and the public Internet. While the second layer switches (aggregation) is mainly responsible for distributing the incoming traffic from the core layer to the lower service layer, they also transmit the aggregation of the dispersed traffic from the downward service layer. In the data center service layer, devices such as a router, Firewall, Ethernet switch, Fiber channel switch, Server load balancing (SLB), and Volume Based Billing/Control (VBB/VBC), are involved [37]. The Firewall permits or denies the incoming network transmissions based on a set of rules for the purpose of guaranteeing the data center`s internal security; it is, a shield for protecting the servers and storage. The SLB distributes the workloads to each server and the VBB/VBC is designed for traffic volume billing.

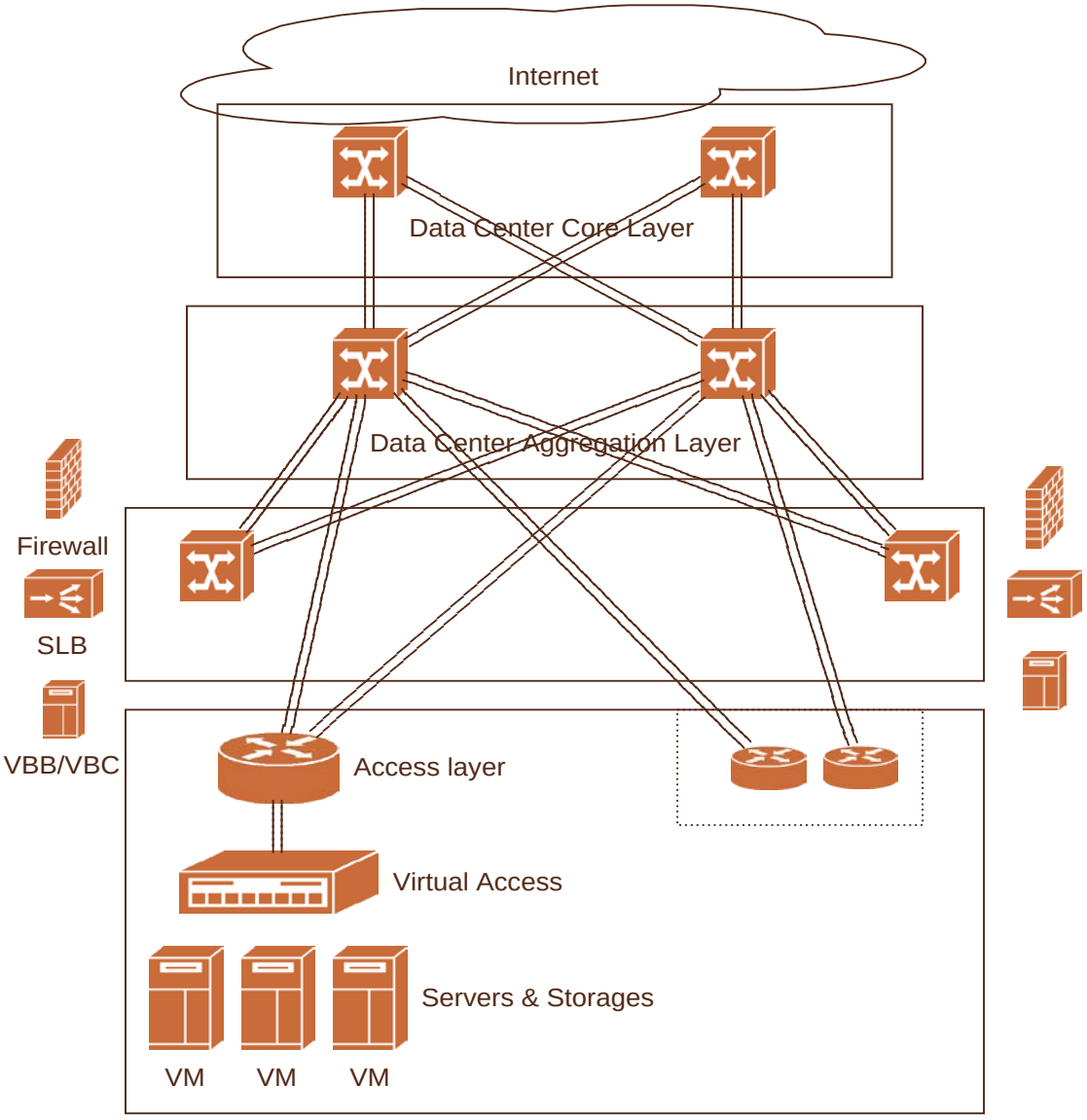


Figure 2.4 - Cloud computing network architecture example

As can be seen in Figure 2.5, network nodes in different layers construct a cloud computing network. The data center uses one or more core routers to connect the Internet and an aggregation layer aggregates the services' layer devices. The services' layer is a complex layer where the application devices such as Firewall, SLB and VBB/VBC are involved. All devices in the services' layer connect to the aggregation layer and are routed by the aggregation device. The access layer includes virtual machines, servers and storage. The access layer offers physical connectivity among servers and the network. This network architecture and configuration are quite complex, and the operation cost is very expensive. End-users are able to submit their service requests flexibly. The network architecture may be different based on each

service. Some cloud users can choose the SLB service and Firewall service while others may choose the billing service. The cloud computing network architecture could be varied based on different types of service provisioning.

2.2 Overview of Data Center Network

A data center is the pool of servers linked together in one place in the cloud. It also has layers of networking that contain routers and switches connected to servers [38]. The increasing number of cloud users creates a challenge in the design of data center networks. This increase makes all communications busy in the data centers. This leads to effects on data centers in terms of performance and energy consumption of the system.

The concept of cloud computing is an immediate extension of many well researched domains such as virtualization, distributed, utility, cluster, and grid computing. According to Google's Whitepaper [39], the five key characteristics of cloud computing are task centric, user centric, intelligence, powerfulness, and programmability. Cloud computing data centers employ virtualization technologies that allow scheduling of workloads on a smaller number of servers that may be better utilized, as different workloads may have different resource utilization footprints and may further differ in their temporal variations.

2.2.1 Performance Trend

According to N. Gorti [40], the demand for computing performance is increasing at an unprecedented speed which is prompted by realistic reasons. The rise of software complexity motivates hardware designers to provide acceptable quality of service (QoS), such as latency, response time, and throughput. Organizations are badly in need of computing capacity for scientific missions with the goal of dealing with ever-challenging large problems at high speeds; for example, the genome sequencing, weather predicting, and molecular dynamics experiments. These tasks require systems equipped with fast processing powers in order to be completed within

an acceptable timeframe. Similarly, cloud computing must be provisioned by gathered computing ability to sufficiently offer diverse services to customers; examples are SaaS, PaaS, and IaaS. Furthermore, with the growing demand for “big data” service, many organizations are trying to adapt their systems to cater to the needs of flexibly manipulating such a service. As a result, the computing demand for data mining is increasing at an unprecedented pace. At the same time, the amount of data generated in such a process is growing in an unpredicted speed. IBM reported [41] that more than 90% of the data in this world has been generated within the past two years.

To satisfy the growing demand for outstanding performance, an increasing amount of transistors has become the trend when manufacturing the cores. On the one hand, chips have become more and more complicated in order to make the cores much more powerful for computing tasks. On the other hand, the number of cores is increasing for each computing node, while the number of computing nodes in a computing platform is also increasing. Figure 2.5 [41] and 2.6 [42] show the consequences of scale-in and scale-out respectively in the IT industry over the last 14 years. In particular, we can observe that the industry has been consistently outperforming Moore's law-based predictions starting from 2000.

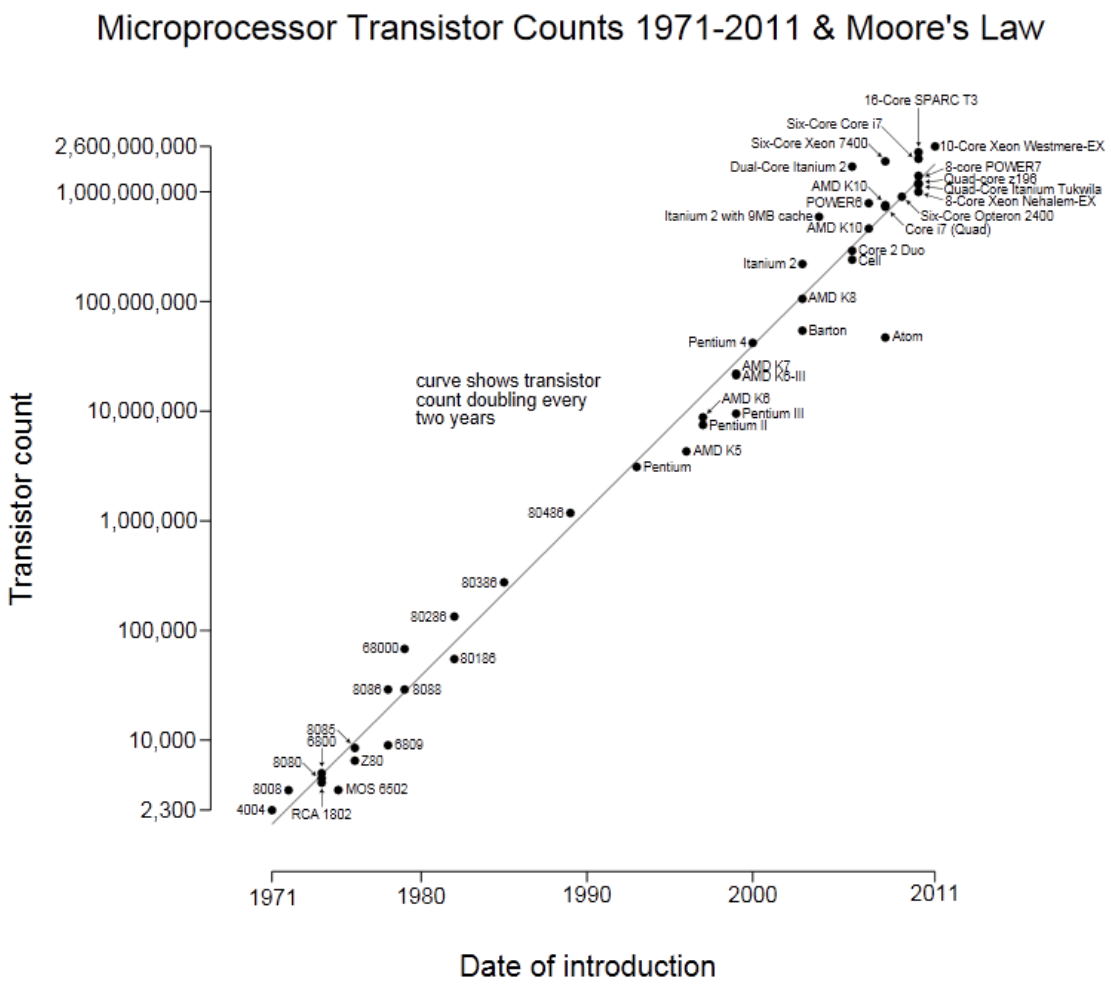


Figure 2.5 – Transistor count from Year 1971 to 2011

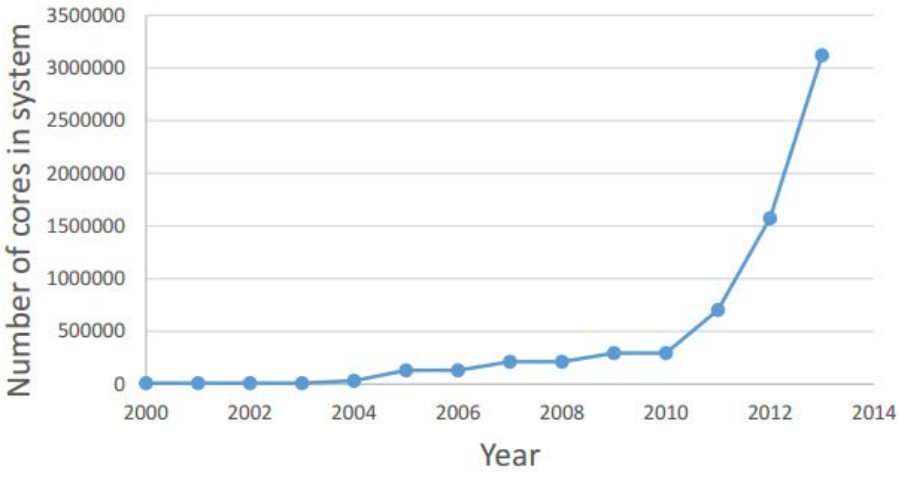


Figure 2.6 – Number of cores in system from Year 2000 to 2014

2.2.2 Power Consumption Trend

As a central issue of datacenters, the energy draw for datacenters can be ranging from kilowatts for a rack of servers to several tens of megawatts for large infrastructure. As stated in [43], some facilities even have power densities more than 100 times that of a typical office building, so that the electricity costs are a dominant operating expense for those facilities and occupy over 10% of the total cost of ownership (TCO) of a datacenter [44]. The cost of power for the global datacenter had already exceeded the cost of the original capital investment by 2012 [45].

Based on the theory of semiconductor scaling [46], power consumption has a decreasing rate of U^2 with each new generation, where U is the reduction factor of voltage per transistor. Even though the new generation of semiconductor manufacturing technology facilitates the reduction of power consumption, the increasing transistor count trend nevertheless pulls the power consumption up anyway, particularly when increasing the computing nodes in a computing platform.

According to a survey conducted by Intel [47], the powering and cooling of servers is the primary factor which limits the growth of the server industry to meet the current global demand. Figure 2.7 shows that 59% of the surveyed group agrees that power consumption is the bottleneck in the development of the server market. In the embedded and personal computing domain, the increase in power consumption leads to decreased battery life and discomfort in device handling. It also necessitates the design of aggressive cooling mechanisms and expensive heat sinks [48-50].

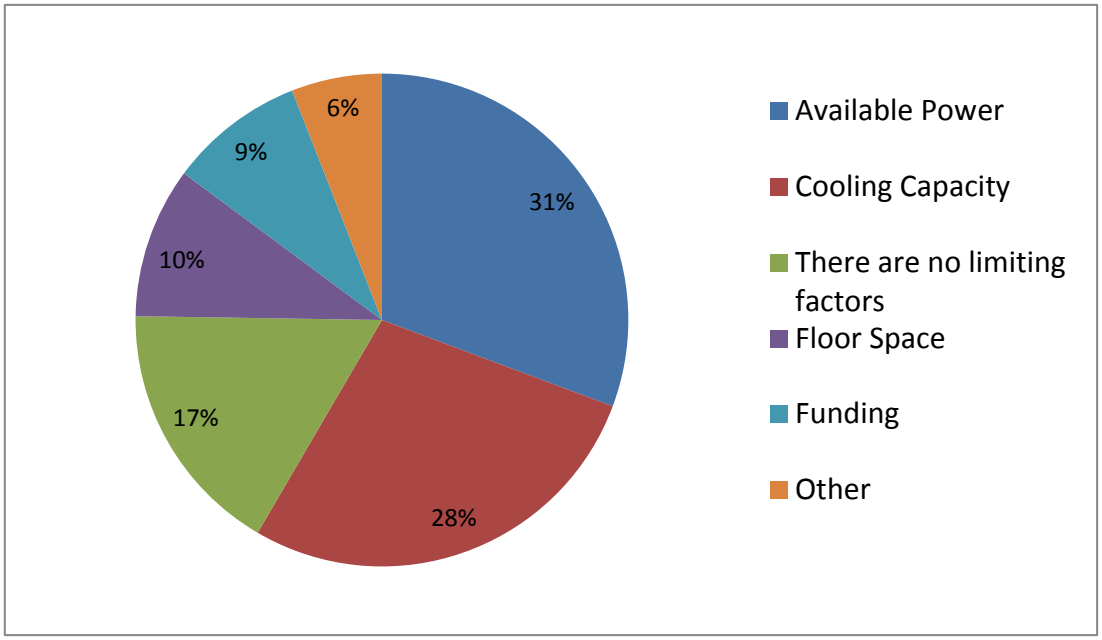


Figure 2.7 – Expenditure of data center components surveyed by Intel

2.3 Data Center Network Topology

A datacenter is the pool of servers linked together in one place in the cloud. It also has layers of network that contains routers and switches connected with servers [50]. The number of cloud users has increased over the last few years, which creates a challenge in the design of datacenter networks. This increase makes all communications busy in the datacenters. This leads to effects on datacenters in performance and energy consumption of the system. This creates a challenge in the design of datacenter architecture and communication protocols. Therefore, there should be topological solutions to increase the performance and reduce energy consumption of the datacenter.

2.3.1 Three-tier

Three-tier DCN is the most commonly known for the current cloud datacenter architecture which contains three layers of switches, including core, aggregate and access switches from the top to the bottom [51]. The core layer allows for multiple aggregation switches to connect together, while aggregation layer switches are responsible for connecting access layer switches between each other. The access layer

contains the connections between the pool of servers and access switches. The core layer connects the layer 2 aggregate switches with the network outside the DCN, and the aggregate switches can be easily added due to their inexpensive character and transitive role, and they are also able to support large number of servers (over 10,000) [1, 51].

The three-tier network topology is easily set up and uses less network components which lower the cost significantly on hardware expenditure [1, 52]. However, the three-tier DCN has limit capacity because of the cost issue which is reflected in link oversubscription, and growing demand for services has increased dramatically in recent years. It lacks scalability, energy efficiency, and cross-sectional bandwidth.

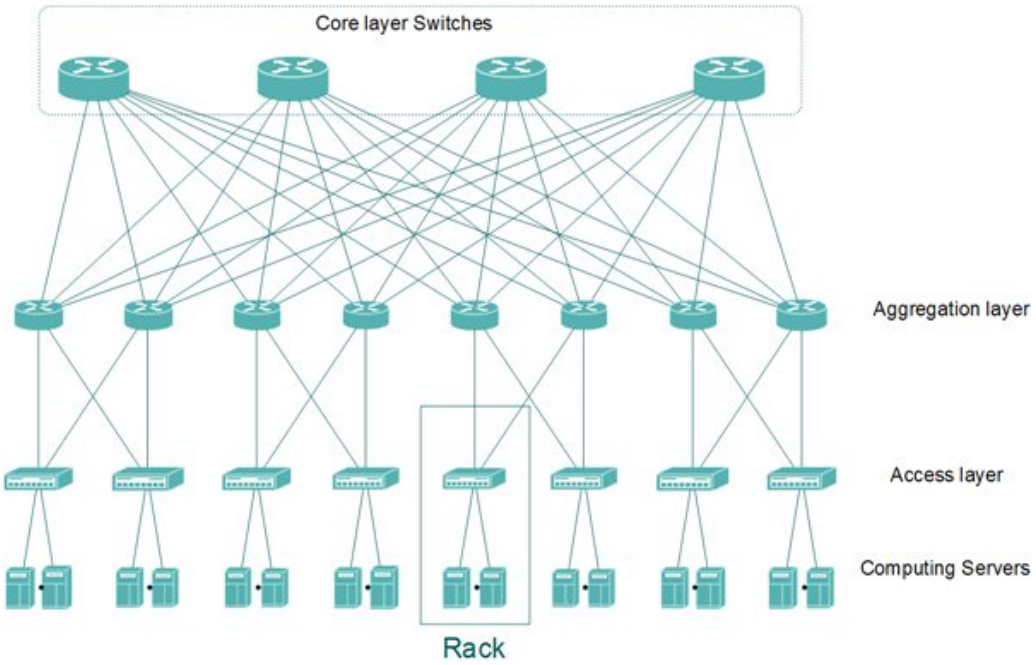


Figure 2.8 – Three-tier DC architecture example

2.3.2 Fat-tree

Fat-tree DCN is the most widely adopted network according to Al-Fares [52]; it follows the hierarchy architecture and contains a core, aggregate and access layers. This structure is composed of k pods, where in each pod there are (k/2) 2 servers, k/2 access layer switches, and k/2 aggregate layer switches. The core layers contain (k/2)2 core switches, where each of the core switches is connected to one aggregate layer

switch in each of the pods. The Fat Tree DCN has advantages in strengthening the ability of oversubscription and cross section bandwidth in contrast to the three-tier DCN. Fat-tree DCN has a larger capacity, but as a result, increases the number of components including both switches and links, which pushes organizations to spend more on maintenance.

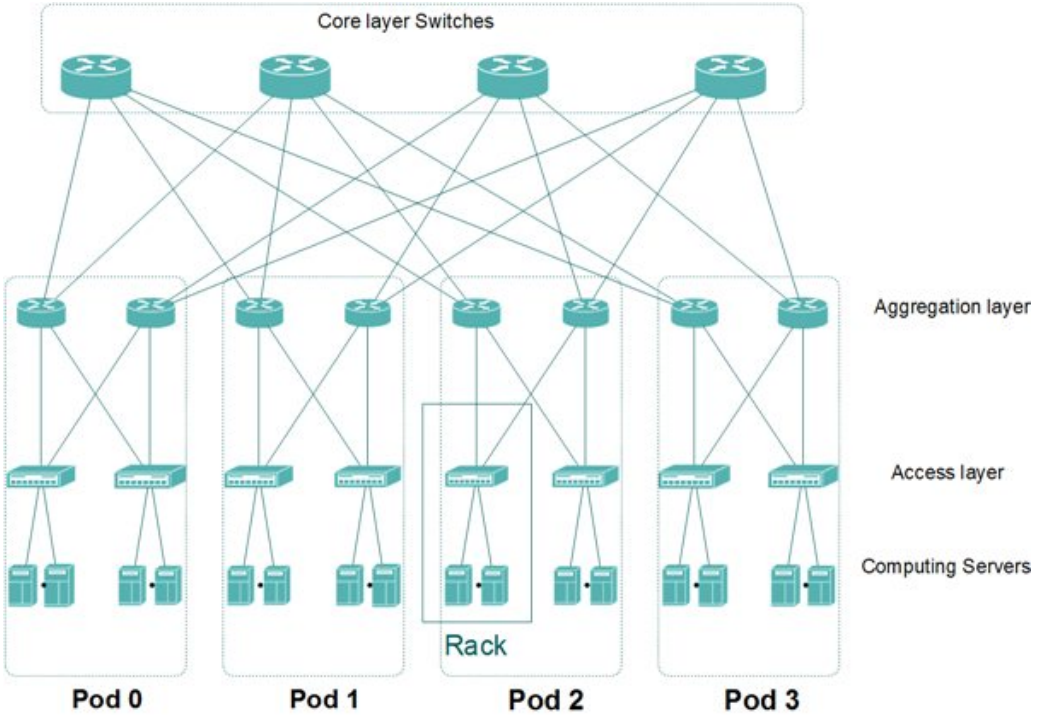


Figure 2.9 – Fat-tree DC architecture example

2.3.3 BCube

BCube [53] topology is a datacenter built inside shipping containers and represents a brand-new DCN shape. It has been proposed to be used as a Modular Datacenter (MDC), which simplifies the installation procedure and implements physical migration, in comparison to conventional datacenters. Datacenter migration facilitates energy saving, because shipping datacenters to regions promotes strategic positioning, and allows for placement close to regions with high service demands. As MDCs are built in sealed containers with a high equipment density, they need to be highly reliable [54]. Furthermore, the equipment has to be moved under control

because failure rate is high when the hardware is not well-protected during the shipping process.

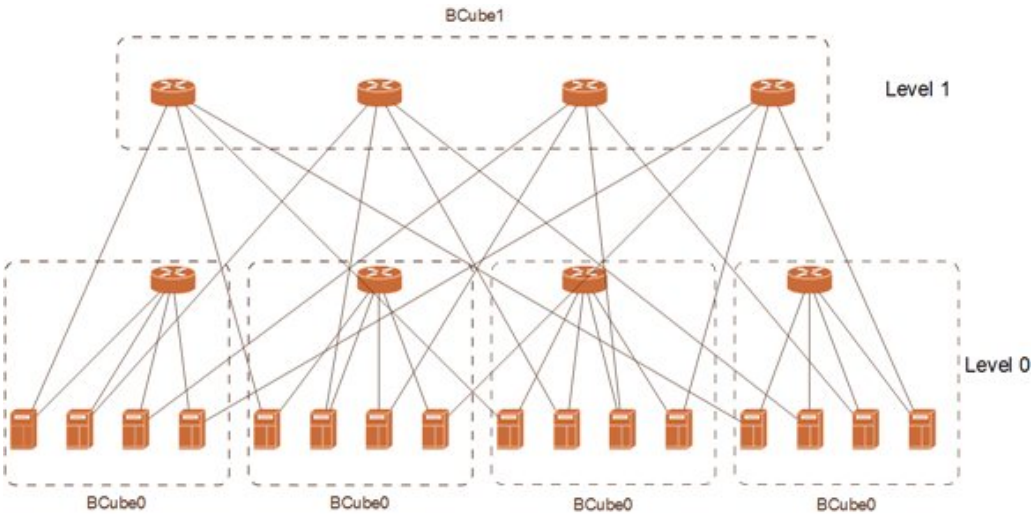


Figure 2.10 – BCube-2 layer DC architecture example

2.3.4 HyperFlatNet

HyperFlatNet is a recursive DCN topology which was proposed by [55]. HyperFlatnet is formed by two layers in which the first layer contains n servers connected by one n-port switch; the second layer consists of n^2 first layers. Hence, the total n^3 servers can be taken as n^2 clusters of n servers. Moreover, different servers can be represented as a n^2 ∗ n matrix where the row and column indexes correspond to the cluster number (i) and the index in the cluster (j). The author proposes a connection algorithm named Linked Clusters Maximization (LCM) to increase the number of directly connected clusters and reduce the number of intermediate hops used to transmit the packet to the destination. A 64-server HyperFlatNet DC architecture is demonstrated in Figure 2.11.

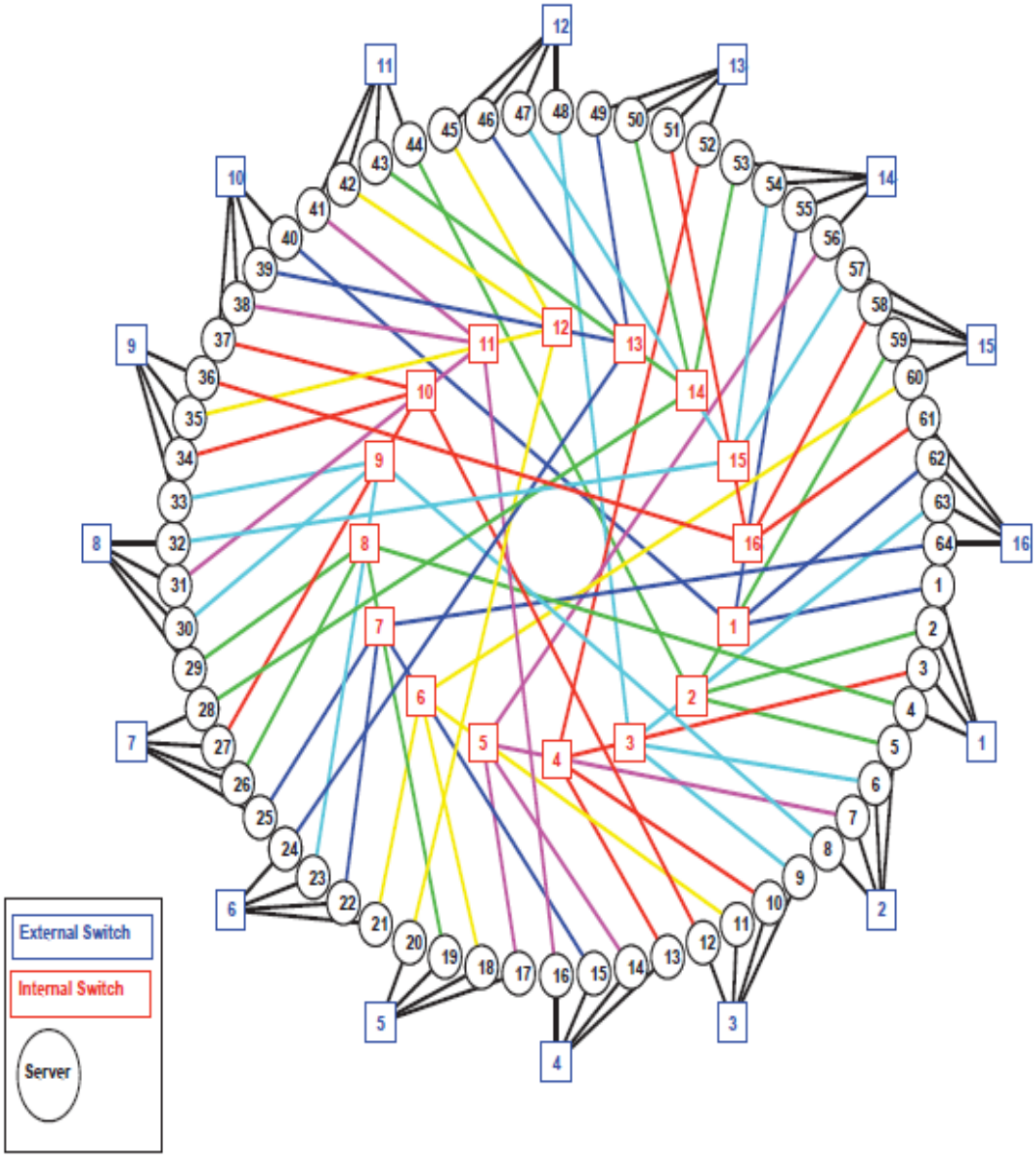


Figure 2.11 – HyperFlatNet DC architecture example

2.3.5 Discussion

Table 2.1 summarizes the four DCN architectures. Both Three-tier and Fat-tree are Clos networks and have high transmission capacity, and BCube and HyperFlatNet are recursive networks which are highly reliable.

DCN type	No. of Layer	No. of Switch type	Architectural type	Characteristics	Cost level
Three-tier	3	3	Clos network, k-ary tree	High transmission capacity	Low cost
Fat-tree	3	3	Clos network, k-ary tree	High transmission capacity	Low cost
BCube	2	2	Recursive network	High equipment density, high reliable	Cost effective
HyperFlatNet	3	2	Recursive network	High performance	Cost effective

Table 2.1 – Comparison of various DCN architectures

2.4 Traffic Load

When the cloud user accesses the services such as instant messaging, content delivery, and social networking by cloud applications from datacenters, a set of servers then generates different levels of workloads that are usually modelled as a sequence of jobs which can be divided into a set of tasks. The tasks are either dependent on the execution of other tasks, or independent. Furthermore, by the nature of grid computing applications such as biological, climate, or financial modelling, the jobs are usually computationally intensive, which needs high workloads to minimize the time required for the computation. The servers are Map-Reduced to accomplish this goal. Usually, the time taken to compute may vary by weeks or months when dealing with a large sequence of jobs.

In cloud computing, the incoming requests generated are always less computationally intensive, but with a strict completion deadline based on the SLA. The majority of cloud computing applications usually generate three types of jobs, which are as follows:

2.4.1 Computationally Intensive Workloads (CIWs)

This kind of job always requires High-Performance Computing (HPC) that aims to solve computation-intensive problems which load computing servers considerably [11], e.g., huge data analysis which needs high computational ability from servers. Furthermore, CIWs also can be clustered into low data transferring (LDT) and high data transferring (HDT); the LDT requires less data transfers on the network so there is a very low probability of causing network congestion. In this case, the idle switches are put into sleep mode which reduces the energy consumed by the datacenter network, while the HDT requires a higher-leveled transmission network than LDT because it tends to produce network congestion. In this case, DVFS takes more effects than DPM to save energy because all network components are in full load when the shutdown of devices is not available.

2.4.2 Data-intensive Workloads (DIWs)

This type of job puts a heavy load on data transfers but produces almost no load at the computing servers [11]. For instance, the loads generated by the applications of video transferring or large file sharing from one simple user requires no computing capacity, but high demand for the interconnection of the DCN, so congestion always occurs through communication links for managing such jobs. Furthermore, the packet drop becomes much more common when the switches are dealing with DIWs. The DIWs always reflect the bottleneck of the DCN that must be deal with high resilient and sustainable DCN architecture to resolve the congestion issue so as to limit the packet drop rate to an acceptable range.

2.4.3 Balanced Workloads (BWs)

BWs are the jobs targeting applications that have both computing and data transfer requirements [11]. The computing servers are in load proportionally to the communication links. The average load on servers therefore equals the average DCN load with this type of job. BWs can model such applications as geographic systems

that take both large data transfers and heavy processing into account. In this thesis, this type of load is being simulated and considers both the server load and the networking load.

2.4.4 Workload specifications in the Simulation

The workloads in the simulation were required executed in two main parts: (a) communicational component and (b) computing components. The communication components are mainly switches and links, which are defined as the amount and the size of data transfers that are performed prior, during, and after the workload execution. The workloads for communicational components are defined in size of bytes. The workload size defines the tasks as divided packets in bytes that are transmitted between servers and switches after the execution of computation in a server.

The computing servers are primarily defined as the amount and the ability of computation in units of packet bytes and server CPU mips respectively, which has to be executed within the limit of a given deadline in seconds. The adoption of the deadline scheme aims at introducing the QoS constraints specified in SLA.

The workload can be specified into several parameters as follows: the size of the workload refers to the amount of bytes being transferred out of servers upon task completion after computing, CPU mips refers to the requirement for completing the computation of the task, deadline refers to the SLA specification for each task, output in this stage means the amount of data in bytes that sent out of the server upon task completion, intercom refers to the amount of data bytes to be transferred to another server, CurrProcRate refers to the current processing rate of the task which is determined by the server, and ExecutedSince refers to the last instance of task execution.

Workload Parameters	Workload Specification
Workload size	amount of data in bytes sent out of the server upon task completion
CompAmount	Amount of computing
CPU mips	computational requirement of the task
Deadline	task deadline
Intercom	amount of data in bytes to be transferred to another server application
CurrProcRate	current processing rate of the task (determinded by the server)
ExecutedSince	last time instance of task execution

Table 2.2 – Workload specification applied in case study

2.5 Data Center Traffic Characteristics

Traffic in the datacenter is commonly flowing in three directions [56]. “North-South” traffic is usually flowing between end-users and servers, which is primarily comprised of traffic that enters and exits the datacenter, and generally contains commands, queries, and specific data either being retrieved or stored. Meanwhile, the “East-West” traffic, flows between DC nodes and applications that never leave the DC. It is primarily composed of communication between applications hosted on physical servers and virtual machines, coupled with virtual machine (VM) to VM, and physical to physical interactions within the DC. As the name implies, “Inter-DC” traffic is largely comprised of resource optimization and disaster recovery requirements between multiple DCs, and between DCs and the private/public cloud.

Cisco’s Global Cloud Index [57] indicates that, the dominant volume of traffic in the DC traverses in an “East-West” direction (76%), followed by “North-South”

traffic (17%), and finally, inter-DC traffic, which is currently only at 7%, but is gradually growing. Moreover, in campus networks, traffic is primarily (> 90%) “North-South” traffic.

2.6 Energy Efficiency of Data Center Network

2.6.1 Dynamic Voltage and Frequency Scaling

Reducing energy consumption is an important research topic and always a challenge for cloud computing organizations. Dynamic voltage and frequency scaling (DVFS) is the most common method in power management to deal with the challenge, where the supply voltage and frequency can be scaled dynamically within a computer component in order to achieve reduced energy consumption. Dynamic Frequency Scaling (DFS) or CPU throttling is mostly a power saving technique in computer architecture where the frequency of a CPU can be automatically adjusted for the purpose of conserving power and reducing the amount of heat produced by the chip. Dynamic frequency scaling by itself can rarely save switching power, whilst Dynamic Voltage Scaling (DVS) is always used in conjunction with DFS in order to conserve the power, because the frequency of a chip is normally run at an operated voltage.

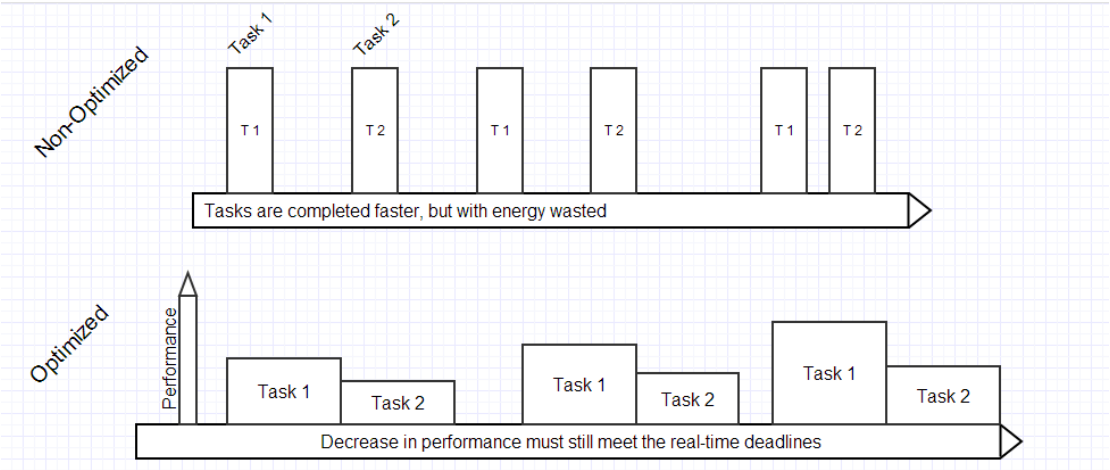


Figure 2.12 – Non-optimized task performance vs. Optimized task performance

The dynamic power (switching power) dissipated per unit of time by a chip is expressed as $\mathrm{C} \times V^2 \times A \times F$, where C is represented as capacitance being switched

per clock cycle, V is the amount of supply voltage in the unit of time, and A reflects the Activity Factor [58] which indicates the number of switching events undergone by the transistors in a chip, and f is the switching frequency [59]. The voltage required for stable operation is determined by the frequency at which the circuit is clocked, and can be reduced if the frequency is also reduced [60]. Increasing voltage (overvolting) and frequency (overclocking) allow an increase in performance such as the task calculation time because the number of instructions a processor can issue in a given amount of time is increased, but that is a power hungry action because the power dissipates in proportion to the square of voltage. Lowering voltage (undervolting) and processor clock frequency (underclocking) is always done with the goal of reducing power while keeping the performance is always a consisting researching topic. For instance, microprocessors such as AMD [61] and Intel [62] allow the CPU speed to be set dynamically.

However, DVFS has been studied for the target of minimizing power consumption. Okuma et al. [63] deployed a few variable voltages to verify that voltage scaling technique is more effective than just stopping the power supply for the components which are idle [63]. Similarly, DeLangen and Juurlink [64] implemented a leakage-aware multiprocessor scheduling algorithm in non-peak performance with a loose task deadline environment to investigate techniques of DVS, and processor shutdown. The results showed that the total energy consumption can be reduced up to 46% for tight deadlines (1.5× the critical path length) and by up to 73% for loose deadlines (8× the critical path length) compared to an approach that only employs DVS. Chen et al. [65] lowered the voltage for non-critical execution tasks without impact the execution time in a mesh network by using the DVFS technique. They reflected that an integrated CPU/communication link voltage scaling method produces much better results rather than only CPU voltage scaling and only link voltage scaling, and they achieved a 13% energy saving over CPU voltage scaling and 17% energy savings over communication link voltage scaling. Wang et al. [66] studied the slack time for non-critical jobs, by extending their execution time and reducing the energy

consumption without increasing the task's execution time. Additionally, the Green Service Level Agreement was considered. Similarly, Kim et al. [67] proposed energy-aware scheduling algorithms based on DVS for bag-of-tasks applications within the limit of deadline requests by application users.

2.6.2 Dynamic Power Management

Dynamic Power Management (DPM) [68] refers to a technique of selective shutdowns of systems for which components are in idle status or underutilized. It is considered to be the most effective method for mitigating the power dissipation, but deploying such a technique also incurs performance degradation due to the frequent shutdowns and wakeups. Therefore, the design of such technique has to be aimed at maximizing the power saving while maintaining performance within acceptable limits.

Power management is a prediction problem; it seeks to forecast whether an idle period will be long enough to compensate for the overhead of power state changes. As reported in [69-70], although a server stays in an idle state, it also consumes around 66% of energy compared to its full load energy consumption, which comes from the fixed component that is not related to the frequency but also consumes power. According to [71], the minimum length of time for a server staying in an idle period is referred as the break-even time (Tbe), and the state transition delay (To) consists of shutdown delay (Tsd) and wake-up delay (Twu); the energy consumed during this period is Eo. The power consumed in working and sleeping states is Pw and Ps. Figure 2.13-(a) represents the working state of the server; Figure 2.13-(b) demonstrates the shutdown state of the server. The break-even time makes energy consumption in both cases equal.

That is to say, the total energy consumed of a server that in working state going through the minimum time length to save power is

$Pw \times Tbe$.

While the total energy consumed of a server that in sleep mode with the minimum time length to save power is

$Eo + Ps \times (Tbe - To)$.

Therefore,

$Pw \times Tbe = Eo + Ps \times (Tbe - To)$.

So that

$Tbe = (Eo - Ps \times To)/ (Pw - Ps)$.

The break-even time has to be larger than the transition delay; therefore,

$Tbe = \max[(Eo - Ps \times To)/(Pw - Ps), To]$.

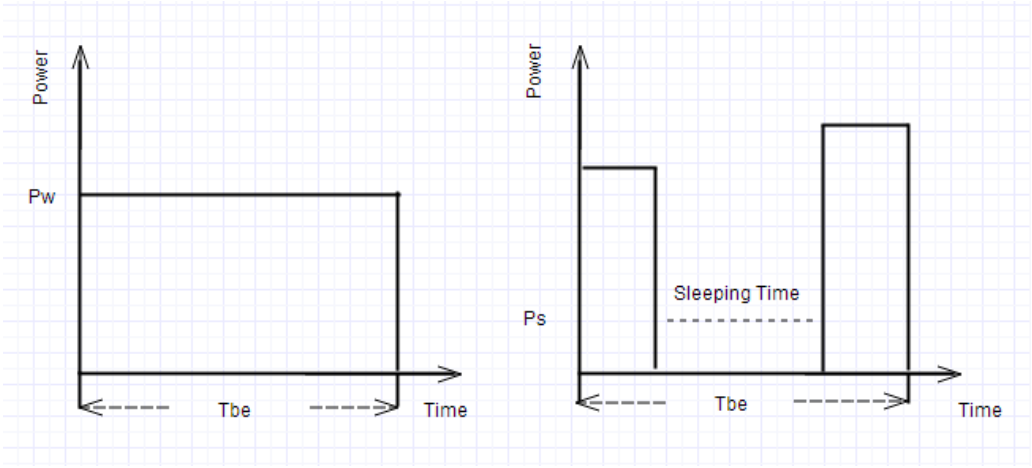


Figure 2.13 - (a) Server with working state, (b) Server with shutdown and wake-up state

DPM policy will only take effect when the server idle time period is longer than Tbe, so the primary goal of any DPM policy is to make the device sleep for at least Tbe. Otherwise, it might cause more power to be consumed than the always 'on' state.

The common DPM policy can be represented as Timeout Policy, Predictive Policy, and a mixture of both policies.

A. Timeout Policies

Timeout is the most conventional policy that is used in DPM and which uses a timeout value of τ. The device is put into sleep mode if it is kept idle for more than τ.

The basic mechanism is that if the device remains idle up to τ, then it should further stay idle for at least Tbe. However, this policy wastes energy while within the value τ.

In [72], the authors adopted a fixed timeout τ that equals the Tbe for accomplishing the target of DPM. Adaptive timeout policies dynamically modify τ based on certain parameters. In [73], Douglis et al. adopted an adaptive timeout policy to dynamically adjust τ according to the ratio of performance delay and sleeping time from the previous idle time period. If the ratio is high, τ is increased, but when the ratio gained is too high, it decreases on the contrary. The maximum and minimum values for τ are considered to forestall the policy being either too aggressive or conservative.

B. Predictive Policies

Another DPM policy applied a predictive algorithm to predict the length of the upcoming idle period. The prediction would generate a decision on whether to put the device into sleep or not, which makes use of the comparison of greater or less than the Tbe metric. In [74], Hwang et al. used an exponential average scheme to predict the upcoming idle period length by taking an exponential average of the predicted and actual lengths of the previous idle period.

Chung et al. [75] proposed another predictive policy that applies an adaptive learning tree to conduct analysis on the basis of prediction confidence level (PCL) to make predictions on the sleeping period against on the Tbe. The tree stores the sequence of idle periods into tree nodes while the PCL is stored in leaf nodes associated with the respective sequence. A finite state machine is applied to update the PCL, and if the prediction is correct, then the FSM updates the PCL as increased, conversely, it recorded PCL as decreased if the prediction was incorrect. The advantage of this scheme is that it has the capability of managing multiple power states.

In [76], Lu proposed a DPM policy which mixed adaptive timeout and predictive schemes. The policy makes the sequence of user requests into clusters named sessions.

The current session length is predicted on the basis of predicted and actual lengths of the previous sessions, which uses a similar scheme to [74]. The session length is then decreased by an adjustment factor if no service requests are receive, instead of immediately issuing a shutdown command. In contrast, if there are requests received, the session length is increased by the same adjustment factor. A shutdown command is issued when the device has been in idle state long enough compared to the predicted session length.

Compared to the timeout policy, the mixture of timeout and predictive policy can solve the problem of wasting energy while waiting for the timeout to expire but it highly depends on the assumption of service requests the user sends. However, both policies only target maximizing the energy savings and do not take performance loss into account.

2.7 Data Center Network Failures

Datacenter networks are subject to power failure, misconfiguration, firmware bugs, topology changes, cable damage, and malicious traffic. Their failure modes are accordingly diverse.

Fog Creek Software suddenly lost access to servers during a regular network reconfiguration maintenance. As stated in [77], a network loop occurred among a set of switches. The gateways controlling access was isolated from the switching management network, producing a brain-split scenario. Neither system was accessible because of a multi-switch BPDU flood. However, the flood should not have happened on the basis of the Bridge Protocol Data Unit (BPDU) standard; this deviation resulted in two hours of total service unavailability.

On April 21, 2011, Amazon Web Services (AWS) suffered service unavailability for more than 12 hours [78], which caused hundreds of high-profile websites to go offline. Moreover, Amazon engineers tried to transfer the traffic away from a main router in eastern US, but the improper routing policy made many its network nodes within the affected zone totally isolated from other nodes within the cluster. Unlike a

normal network interruption, this change disconnected both the primary and secondary networks simultaneously, leaving the affected nodes completely isolated from one another. This failure also caused an outage in Amazon's RDS (Relational Database Service). When one "Availability Zone" (AZ) fails, RDS is designed to failover to a different AZ; however, 2.5% of multi-AZ databases in the eastern U.S. failed to failover because of a bug in the failover protocol. This correlated failure caused widespread outages for clients relying on AWS. For example, Heroku reported between 16 and 60 hours of unavailability for its users' databases.

To implement failures concretely, each type of fault can be presented as component failures in a DC, where component failure is classified by link failure, server failure, rack failure, and switch failure. Figure 2.14 depicts the example of the fault types in a BCube DCN, where with under-redundant connections, a link failure cannot effectively stop the traffic between source node and destination node, and traffic can be still switched from an alternative route. A server failure will immediately isolate the server from the rest of the network. A switch failure can cause significant influence on the network, as shown in Figure 2.14. The minimum hop count among servers in the second Cube will be increased from two to six which will largely increase the average network latency. On the other hand, the Cube failure causes an unimagined consequence because eight links will be totally disconnected which involves a total failure of a switch.

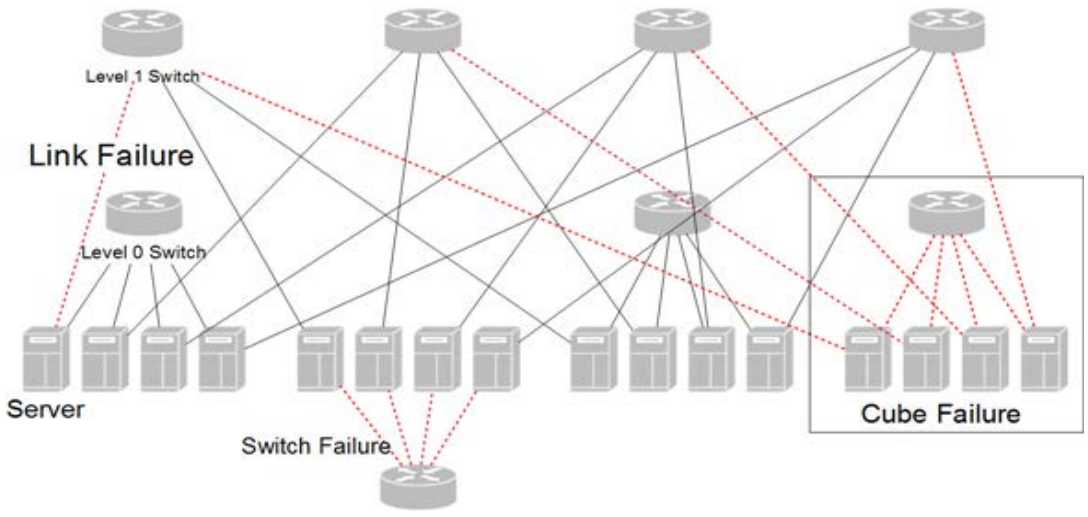


Figure 2.14 – DC component failure example

Chapter 3 Topological Modelling and Metrics

The modelling and simulation techniques are used to answer my research question. Both techniques are commonly used in networking due to the difficulties of experimenting with real configurations. The DCN architectures are modelled and studied by using CloudNetSim++ (an extension of OMNeT++) cloud simulator, as well as Gephi network analysis toolkit.

3.1 Network Modelling and Analysis Tools

3.1.1 CloudNetSim++ simulator

OMNeT++ [79] is an extensible, modular, component-based C++ simulation library and framework, with strong GUI support primarily for building communicational network simulation. It implements C++ language and offers powerful simulation class libraries. In OMNeT++, a network model consists of nested entities in hierarchical order which called modules. Simple modules (e.g., links, server) can be grouped into a compound module, and a network module is normally composed of a mixture of compound modules and simple modules.

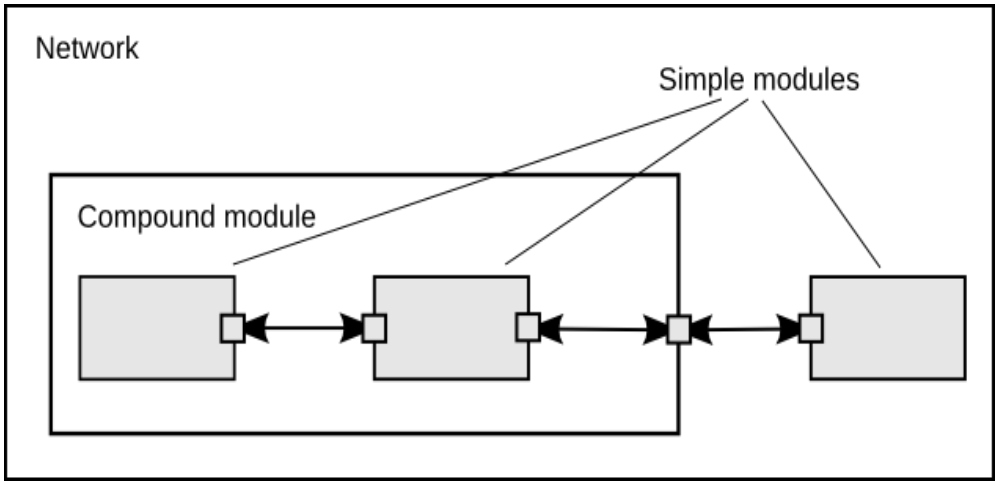


Figure 3.1 - Simple and compound modules [80]

Modules can communicate via message passing which is the central communication mechanism of OMNeT++, where the messages contain strictly complicated data structures. Messages can be sent either directly to destination addresses from source addresses or along a predefined path, through gates and edges, which can be assigned properties like bandwidth, delay and length, and error rate. Modules can have parameters which are used to customize module behavior, create flexible model topologies and for module communication as shared variables. The user must provide the lowest level module in the hierarchy, containing the algorithms in the model.

CloudNetSim++ [81] is a modeling and simulation toolkit to facilitate simulation of distributed datacenter architectures, energy models, and high speed data centers' communication network. CloudNetSim++ [82] is the first cloud computing simulator that uses real network physical characteristics to model distributed datacenters. CloudNetSim++ provides a generic framework that allows users to define Service Level Agreement (SLA) policy, and schedule algorithms and modules with ease for different components of datacenters without worrying about low level details. The CloudNetSim++ is designed to allow researchers to incorporate their custom protocols and applications, and to analyze under-realistic datacenter architectures with network traffic patterns. CloudNetSim++ works with the INET framework, on the basis of OMNeT++. Each datacenter can be regarded as a network module which is composed of compound modules and simple modules, so messages can be transmitted via modules inside a datacenter, between datacenters, or with clients.

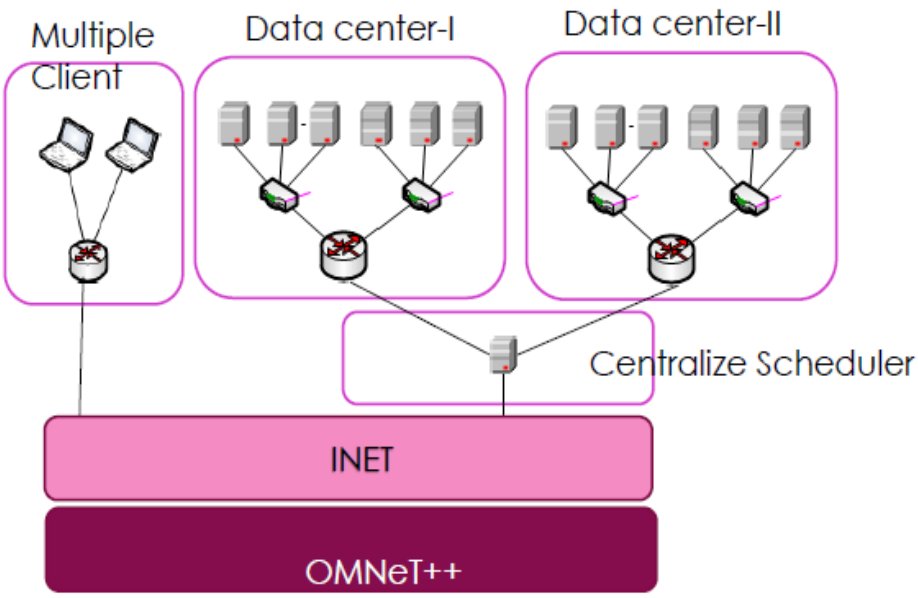


Figure 3.2 – CloudNetSim++ simulator higher architecture [83]

3.1.2 Gephi Network Analysis Tool

Gephi is an open-source network analysis and visualization software package written in Java on the NetBeans platform,[84] initially developed by students of the University of Technology of Compiègne (UTC)[85] in France. It adopts a 3D render engine to display large networks in real-time and to speed up the exploration of such networks. The flexible and multi-task architecture brings new possibilities to work with complex data sets and produces valuable visual results. It provides easy and broad access to network data and allows for spatializing, filtering, navigating, manipulating and clustering. Gephi [86] is an interactive visualization and exploration platform for all kinds of networks and complex systems, with dynamic and hierarchical graphs. It gives researchers the ability to see its data from a new angle. In Gephi, a network consists of two components: a list of the vertices (nodes, in Gephi) composing the network, and a list of the relations (edges, in Gephi). Gephi provides the metrics of robustness of a network such as node degree, betweenness centrality, closeness, diameter, clustering coefficient, PageRank, community detection (Modularity), random generators, and shortest path.

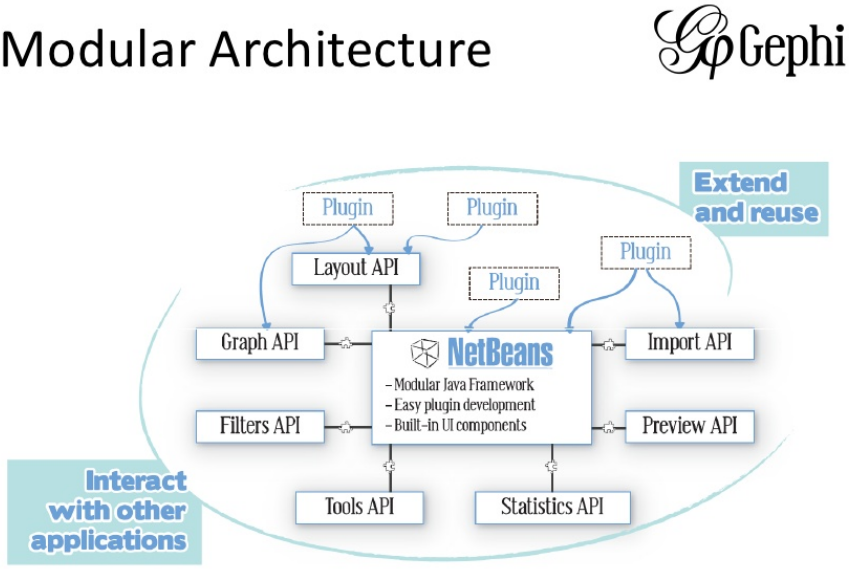


Figure 3.3 – Gephi network analysis tool: Modular architecture [101]

3.2 DCN architectural models

In this work, each DCN implemented a 64-server architecture, for the sake of being able to compare topologies. The link error rate was set as 0.8%, and various link bandwidths were adopted in the range from 1Gigabit/second to 100Gigabit/second. There was always one switch that was responsible for connecting outside the datacenter, for example, the core switch in Three-tier and Fat-tree; the upper layer switch in BCube; and the external switch in HyperFlatNet.

3.2.1 Three-tier

The conventional Three-tier topology follows a tree-based architecture which has three layers, i.e., core layer, aggregation layer, and access layer. The core switches are responsible for connecting the network outside the DC, and also the aggregation layer. The aggregation layer switches the link between the upper core layer switches and the access routers, where access routers are normally placed inside the Rack to connect a set of servers.

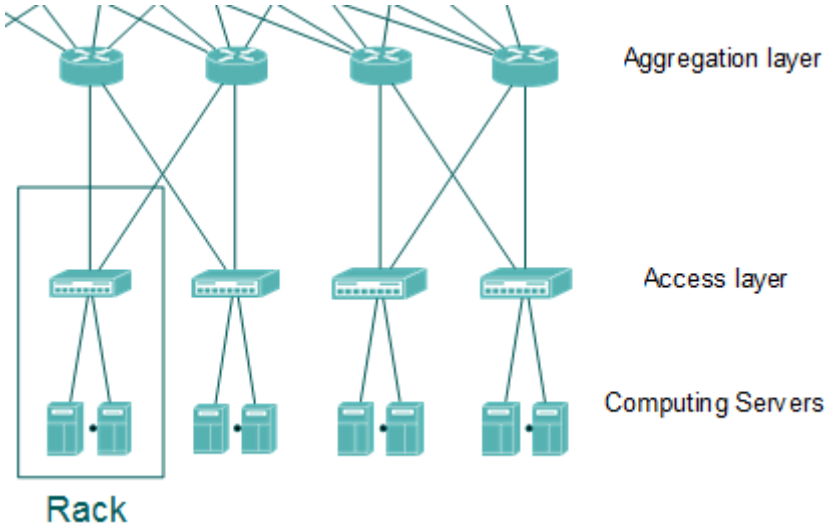


Figure 3.4 – Three-tier Rack example

The conventional Three-tier DC belongs to the type of "stable" architecture, where "stable" equates to more connections in comparison to the other tree-based topology, the Fat-tree. The prominent difference with Fat-tree from a topological aspect is that the number of connections between the core layer and the aggregation layer is doubled and follows a "connect all-to-all" pattern as shown in Figure 3.5.

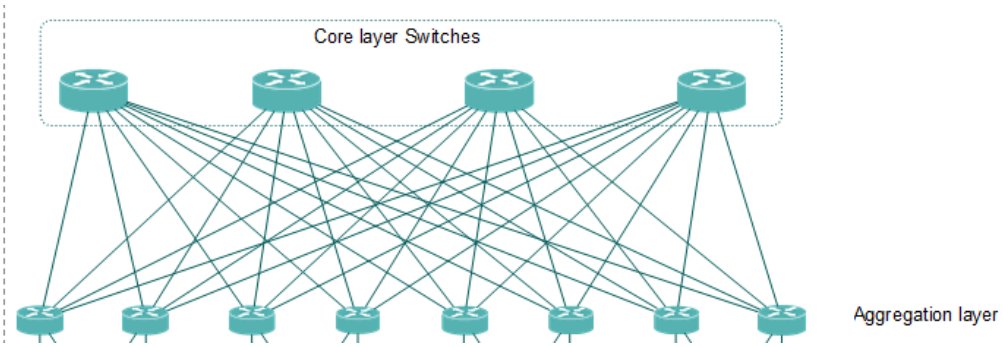


Figure 3.5 - Three-tier model – more robust with "connect all-to-all"

Each core switch connects all aggregation switches so that the in-between links are fully connected. This characteristic demonstrates a highly average nodal degree whose features are more robust as described in the following robustness metric section. Moreover, the interconnections between aggregation layer and access layer for Three-tier DCN are diverse; in this thesis, the same connections as with Fat-tree architecture were implemented.

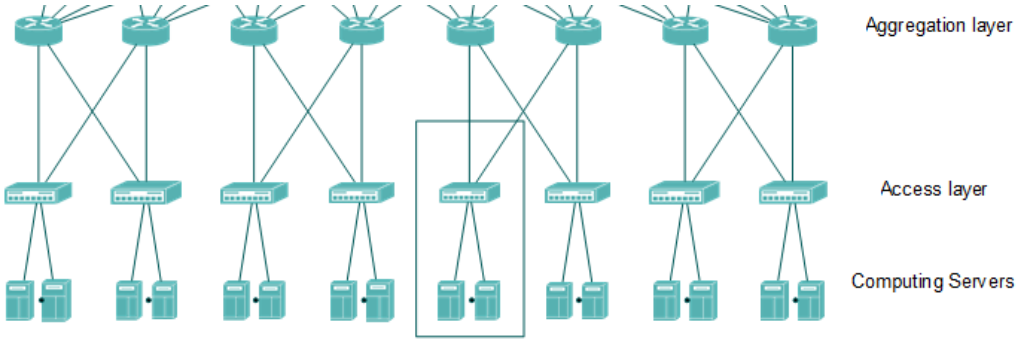


Figure 3.6 - Three-tier: same connections with Fat-tree which between aggregation layer and access layer

The tree-based DCN architecture can be modelled as two modules. The Three-tier model is composed of a Rack module and an internal Three-tier module, where the Rack module consists of one access router with a set of servers connected, and the internal Three-tier module consists of the internal DC connections. Each of core switches links with each of the aggregation switches, while the connections between the aggregation layer and the access layer are identical with the one in Fat-tree.

3.2.2 Fat- tree

According to the architecture of Fat-tree DC, the tree-based architecture which highly relies on the server Rack assembled can be modelled as sectionalized modules, and the rack can be regarded as an entirety which involves servers placed inside with the edge router connected. The rest of the architecture can be referred to as the interconnections among core switches, aggregation switches, and the Racks.

Hence, the entire Fat-tree DCN can be formed by Fat-tree internal modules and Rack modules, where the computing servers and edge routers are included inside a Rack module. For each Rack, there are “N” computing servers connected to only one edge router which is connected with the aggregation layer outside the Rack. In a 64-server Fat-tree architecture, four core switches are included and there are four pods in total, where for each pod, there are two Racks (each Rack has one edge router), and two aggregation switches. Hence, there are four core switches connected with eight aggregation switches, however, the number of links which connects between the core layer and aggregation layer is half reduced in contrast with the conventional

Three-tier architecture so as to mitigate the link oversubscription issue. Compared to the “connect all-to-all” pattern, each core switch in Fat-tree only connects half the number of aggregation switches, where the first half of the numbers of core switches connect to the first aggregation switch in each pod, and the second half of the numbers of core switches connect to the second aggregation switch in each pod, shown in the following figure.

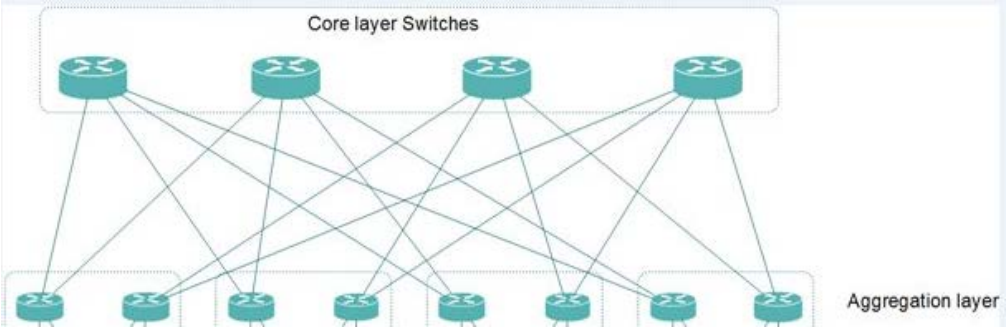


Figure 3.7 – Fat-tree: interconnection of core and aggregation layer

The connection between aggregation layer and access layer complies with the “connect all-to-all” pattern where each aggregation switch is connected to each access switch in each pod.

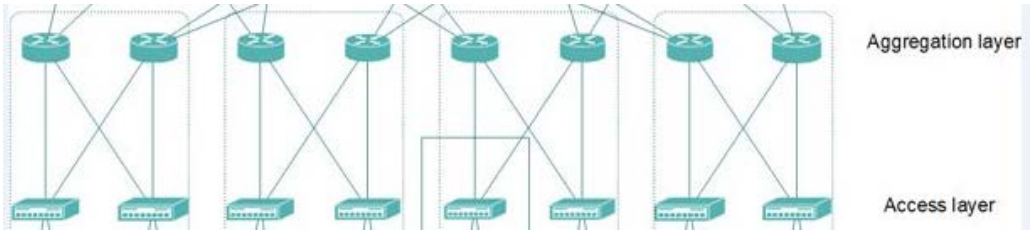


Figure 3.8 – Fat-tree: interconnection of aggregation and access layer

In each Rack, there are 8 computing servers being connected by 1 access router.

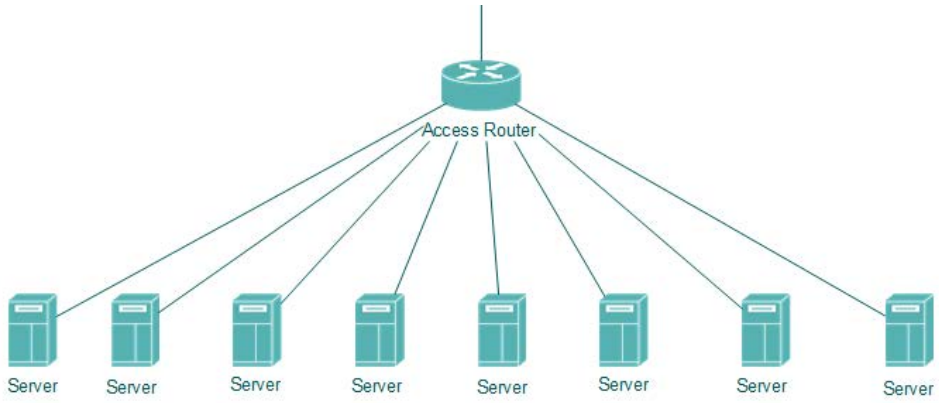


Figure 3.9 – Fat-tree: interconnection of access router and servers

3.2.3 BCube architecture modelling

BCube DC architecture complies with the shipping container principle that applies a “servers place in a Cube” method to achieve the goal of “portable”. Similarly, a Cube module is much like a Rack module, where a Level 0 switch is connected with a set of servers. In fact, Cubes are always considered as Level 0 (L0) of a BCube DC, whereas in an upper layer, Level 1 (L1) switches are placed above the first level. Normally, a typical BCube consists of the same number of L0 switches and L1 switches. The number of Cubes is considered to be the same as the number of switch ports. Hence, in a 64-server BCube with two layers, eight servers are connected with each L0 switch, where there are eight for both L0 switches and L1 switches. Due to BCube is being a server-centric architecture, instead of a L1 switch being connected to a L0 switch, the *i*th L1 switch connects to the *i*th server in each Cube.

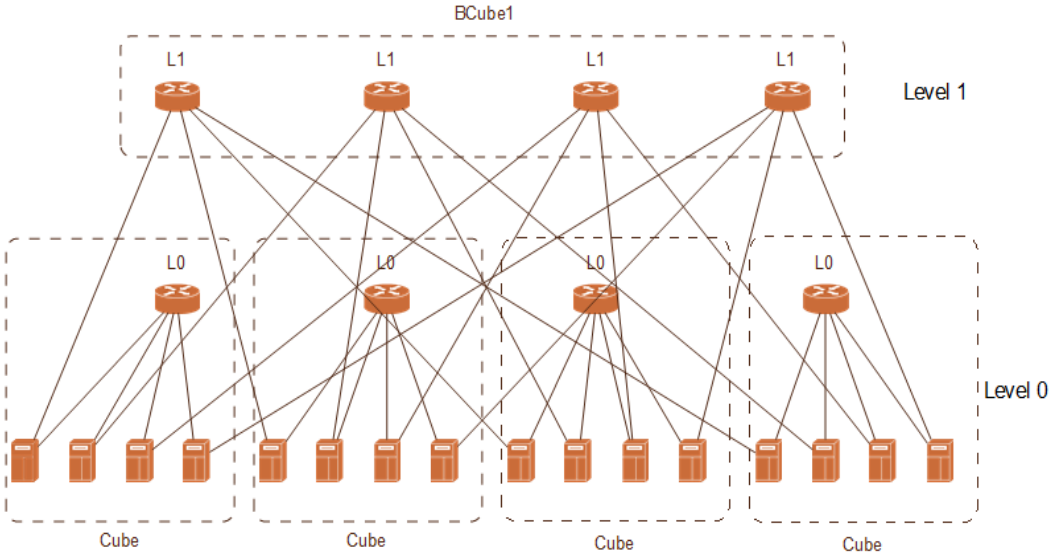


Figure 3.10 – BCube-2 layers interconnection

3.2.4 HyperFlatNet architecture

According to the property of HyperFlatNet, HyperFlatnet has two layers where the first layer has n servers and only one n-port switch, and the second layer consists of n^2 first layers. Thus, there are n^3 servers in total which are connected by n^2

groups of switches; a group here can be considered as a cluster. The hyperFlatNet modelling can be deployed by a matrix function. n^3 servers can be treated as a $n^2 * n$ matrix. The row and column indexes correspond to the cluster number and the index in the cluster, respectively. In this way, every server would be assigned a specific index for the use of DCN connections.

Each server can be connected by the Linked Clusters Maximization (LCM) algorithm, where the matrix denotes as L, which is generated as

$$\forall \mathrm{i} \in \{1..n^2\},$$
$$\forall \mathrm{j} \in \{1..n\},$$

Then the matrix can be completed as

$$\forall \mathrm{i} \in \{1..n^2\},$$
$$\forall \mathrm{j} \in \{2..n\},$$
$$\mathrm{L}(i,j) = mod(L(i-1,j)+1, n^2). \quad (3.1)$$

L_1 refers as the first line of the matrix L, the LCM is shown as follows,

Algorithm 1 Linked Clusters Maximization algorithm

```
procedure LCM(n)
    L_1 = [1]
    for i = 2..n do
        D=[]
        for j = 1..n^2 do
            L_1(i) = L_1(i − 1) + j
            D(j)=ConnectedClusters(i,L_1)
            if D(j) = i(i − 1) then
                Break
            end if
        end for
        j_selected = argmax(D)
        L_1(i) = L_1(i − 1) + j_selected
    end for
end procedure

function ConnectedClusters(p,L_1)
    LC = []
    for i = p..1 do
        for j = 1..i − 1 do
            LC = [LC  L_1(i) − L_1(i − j)]
        end for
    end for
    LC = unique(mod([LC  n^2 − LC], n^2))
    return (Length(LC)) ;
end function
```

Figure 3.11 – Linked Clusters Maximization Algorithm [55]

3.3 Energy consumption modelling

In this thesis, we deployed an energy model by using the DVFS technique and DPF fixed timeout policy together as shown in the following algorithm, and the fixed timeout scheme was used as DPM policy. As Figure 3.12 [87] shows, when the servers are underloaded after a period of execution time at full load, the DVFS technique is adopted to save energy, while if the servers are transferred into idle state, DPM policy can be implemented.

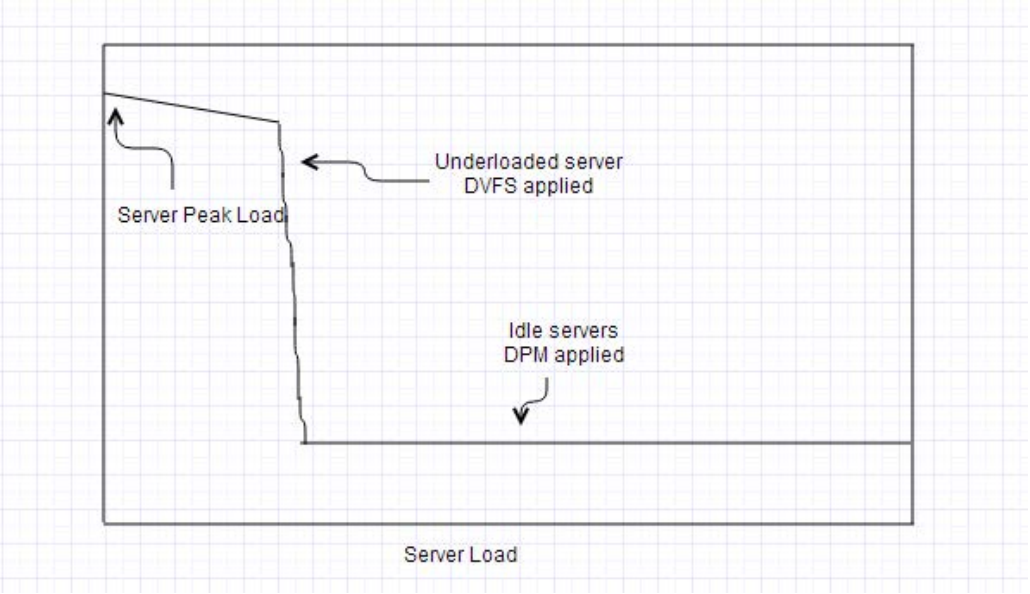


Figure 3.12 – Server load vs. DVFS/DPM applied

3.3.1 Energy Consumption Measurement Algorithm

In this thesis, energy consumption algorithm is deployed by DVFS technique and DPM together. In DVFS [88], chip switching power decreases proportionally to $V^2 * f$, where V is voltage and f is switching frequency. The core principle is that the average power consumed has a cubic relationship with the CPU frequency, moreover, the power consumption for the components which are not related to f that remains fixed, such as bus, memory, and disk. Therefore, the server power consumption can be stated as follows,

$$P = P_{fix} + P_f \times f^3 \tag{3.2}$$

Where P_{fix} is the fixed power consumption by components not linked with frequency such as the bus, memory, and disk, P_f is CPU power consumption linked with frequency.

On the other hand, according to [11], the total energy consumption for the DCN can be divided into three main portions: computing energy by servers, communication energy consumed by links and network equipment operations, and the power consumed by infrastructure for supporting datacenters (e.g., cooling/air conditioning

system). Only a fraction of energy consumption has been delivered to the computing server directly, another considerable portion of the energy being consumed for maintaining interconnection links and network equipment operations. The power consumption for switches makes up a great proportion to the overall DCN power consumption. The researcher set up three scenarios with the GreenCloud simulator to compare the detailed network component energy consumed among Two-tier DCN, Three-tier DCN and Three-tier high-speed DCN. The results showed that the energy consumption for the switches accounted for 26.54%, 30.27% and 30.98% respectively, which take up around 1/3 of the total energy. As stated in [11] the energy consumed by a switch can be expressed as:

$$P_{switch} = P_{chassis} + n_{linecard} * P_{linecard} + \sum_{i=0}^{configs} nports_{configs_i} * P_{configs_i} \quad (3.3)$$

Where $P_{chassis}$ is the power consumed by switch hardware, $P_{linecard}$ represents the line card power consumption with no ports turned on, $n_{linecard}$ represents the number of cards plugged into a switch, $P_{configs_i}$ is related to the power consumed for a port running at rate i, $P_{chassis}$ and $P_{linecard}$ are fixed due to the operation of a switch, so in this equation, only $P_{configs_i}$ is dependent on transmission rate i which is proportional to the overall power consumption of the switch. In other words, the transmission workload directly influences the total energy consumption of a switch; more tasks go through a switch, more energy a switch consumed.

On the other hand, the DPM model allows the server power to be shut down whenever the servers are in zero load; this mechanism is presented in the following algorithm, in addition to the DVFS technique.

To make the model accurate, some data was notarized first as prerequisites. A typical cloud server was used as the model, running an Intel Xeon processor [89] with a nominal energy consumption rate fixed as 301W / hour and the CPU nominal mips to be 2000, and around 171 W / hour allocated for other peripheral devices. 171w/h is

consumed by memory modules, disks, I/O resources, and other peripherals in an acceptable state. Then, the power consumption linearly increased with the level of CPU load.

CPU Nominal Energy Consumption Rate	301w/hour
CPU Peak Load Energy Consumption Rate	130w/hour
Power Rate for Component not linked with frequency (idle status)	171w/hour
CPU Nominal Mips	2000

Table 3.1 - Server Energy Consumption Rate Specification

Algorithm 2: Energy Model algorithm application

/* Compute idle server energy consumption */
Idle Server Energy Consumption Rate = Nominal Energy Consumption Rate*2/3;
/*mission load is calculated according to CPU metric: Mips (Million instructions per second)*/
Current Load = Current Mips / Nominal Mips;
/* frequency component */
CPU Frequency = Current Load

/* if DPM is enabled no energy is consumed with zero load */
if Current Load == 0 && DPM -> enabled then
Current Consumption Rate = 0;
return;
else if DVFS model -> enabled **then**
/* if DVFS is enabled energy consumed is scaled with the frequency */
Current Consumption Rate = Idle Consumption + Nominal Rate * f*f*f / 3;
return;
end if
end if
/* Compute load dependent energy consumption component */
Load Component Energy Consumption Rate = (Nominal Energy Consumption Rate – Idle Server Energy Consumption) * Current Load;
Current Consumption Rate = Idle Server Energy Consumption Rate + Load Component Energy Consumption Rate;
/*Energy consumption calculation*/
Energy Consumption = a period of time from last update* Current Consumption Rate;

3.4 Network Performance Measurement

Based on the existing DCNs infrastructure, there are possibilities to conduct topological change to improve the DCNs performance. The communication components for networking such as the switches and links directly influence the network performance, so as to the Cloud infrastructure. Normally, due to the geographical distribution character, the links which connect each data center are always equipped with relative long distance, which link delay cannot be neglected as a performance metric. According to [90], the latency or packet delivery time indicates the time spent from the first bit sent from transmitter to the last bit received by the receiver, which is composed of packet transmission time and link propagation delay. The average packet delay can be calculated as the followed, where D_{avg} refers to the average packet delay and represents to the number of packets received, d_i refers to the delay of the packet i.

$$D_{avg} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{3.4}$$

On the other hand, when a data stream is transmitted over a communication channel, it exists possibilities of the number of received bits be altered due to the link noise, interference, etc. [91], this leads to a probability of packet dropping. In addition, the packets also have the probability of being dropped when a large data stream traffic pass the switch especially when the switch load burdens beyond its frame capacity, the packets are dropped automatically by switch queue mechanism. Furthermore, the simulation will adopt this model in the situation of traffic are forced to be transferred to alternative routes due to the link failures so that alternative switches will bear the extra weight of load.

Algorithm 3: Packet dropping algorithm – DropTailQueue mechanisim

```
/** the algorithm is divided into two parts: enqueue and dequeue. **/
Enqueue:
if queue = Empty then
    return NULL;
else
    if frameCapacity && queue.length() >= frameCapacity then
        Queue -> is full, dropping the packet;
```

```
            return msg;
        else
            queue->insert(msg);
            emit -> queueLengthSignal, queue.length();
            return NULL;
        end if
end if
Dequeue:
if queue -> empty then
    return NULL;
else
    cMessage *msg = (cMessage *)queue -> pop();
    /** unlinks and returns the front element in the queue. If the queue was empty,
    error is thrown. **/
     emit -> queueLengthSignal, queue.length();
     return msg;
end if
```

The average network throughput can be calculated as the followed, where T_{avg} refers to the average throughput in the network, $p_i \in [0,1]$ while 0 reflects the loss of packet i and 1 indicates the receipt of packet i, δ_i refers to the size of packet in bits and d_i represents the delay of the packet, n as the number of packets received.

$$T_{avg} = \frac{\sum_{i=1}^{n}(p_i \times \delta_i)}{\sum_{i=1}^{n} d_i} \quad (3.5)$$

3.5 Topological Metrics

The researcher used graph theory in this thesis to evaluate different topologies, especially for the aim of network analysis. The prerequisites included the most basic definitions of a graph, network interpretations, and robustness metrics that prioritized used in the network graph.

A graph [92] can be represented as a pair G = (V, E) of sets assuming that $V \cap E = \emptyset$ where V is depicted as the vertex. The elements of vertices (or nodes in the case of a network), and E represents a set of edges (or links). To draw a graph, the only essential information needed is which pair of vertices forms an edge.

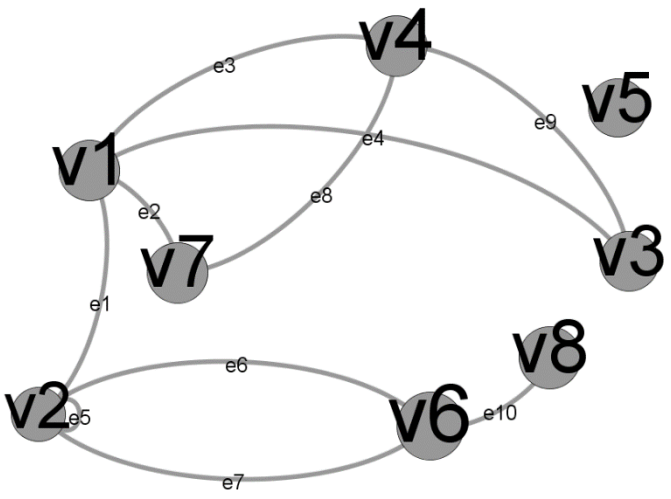


Figure 3.13 - The graph on V = {v1…v8} with edge set E = {e1…e10} = {(v1, v2), (v1, v3), (v1, v4), (v1, v7), (v2, v6), (v3, v4), (v4, v7), (v6, v8), (v2, v2)}

From the Figure above, we have the following terminologies, as example:

• v4 and v3 are end vertices of e9.

• e5 is a loop.

• e6 and e7 are parallel because they have the same end vertices.

• The graph is not simple because it has parallel edges or loops.

• e4 and e3 are adjacent.

• v1 and v4 are adjacent.

• The degree of v8 is 1 so it is a pendant vertex.

• e10 is a pendant edge.

• The degree of v2 is 4.

• The degree of v3 is 2.

• The degree of v5 is 0 so it is an isolated vertex.

In a graph G, the degree of a vertex v is the number of edges at v, as the consequence, it is equal to the number of its neighbors.

The minimum degree of the vertices [92] is denoted as $\delta(G) = min\{d(v)|v\epsilon V\}$,

In particularly, δ (G) = 0 if there is an isolated vertex in G. Similarly, we denote Δ (G) as the maximum degree of vertices in G, $\Delta(G) = max\{d(v)|v\epsilon V\}$. In this case, δ (G) = 0 and Δ (G) = 4. We calculate the average degree of a vertex v as the following formula,

$$d(G) = \frac{1}{|V|}\sum_{v\in V} d(v) \tag{3.6}$$

Where, $\delta(G) \leq d(G) \leq \Delta(G)$.

A path is a $\neg\emptyset$ graph $P = (V, E)$ of the form of

$V = \{v_0, v_1, v_2, v_3, \dots, v_k\}$ $E = \{v_0v_1, v_1v_2, v_2v_3, \dots, v_{k-1}v_k\}$,

The vertices v_0 and v_k are connected by path P, where v_0 and v_k are called its "end vertices". The number of edges of a path indicates the Path Length k (denoted by P^k), we usually refer to a path as a natural sequence of its vertices, a path P from v_0 to v_k.

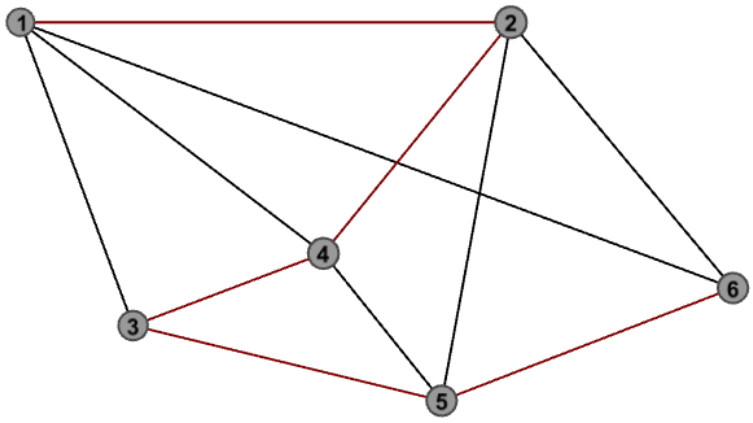


Figure 3.14 – Example – 6 nodes network

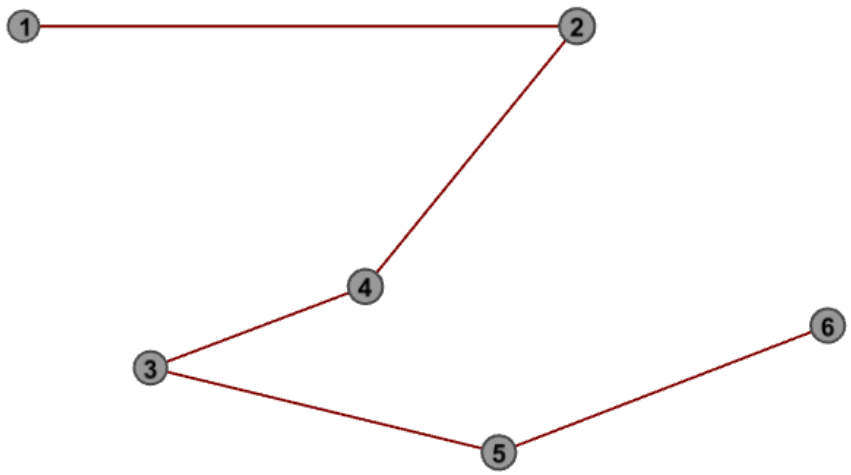


Figure 3.15 - Example - A path P^k = 5 in G

A graph which does not contain any cycles is called a forest; the components of the forest are called trees. The vertices of degree 1 (a pendant vertex) are called leafs in one tree, while each tree has at least one or more leafs, for example, the end vertices of a longest path. As shown in Figure 3.16, one of the longest paths, P^k, is oriented from Leaf (0) to Leaf (k), and there are 10 longest paths in total. As stated in the previous section, Fat-tree and Three-tier are called Tree-based Topologies; the longest path which is mostly dominated by the switch layer must be from one node to the leaf – the computing server.

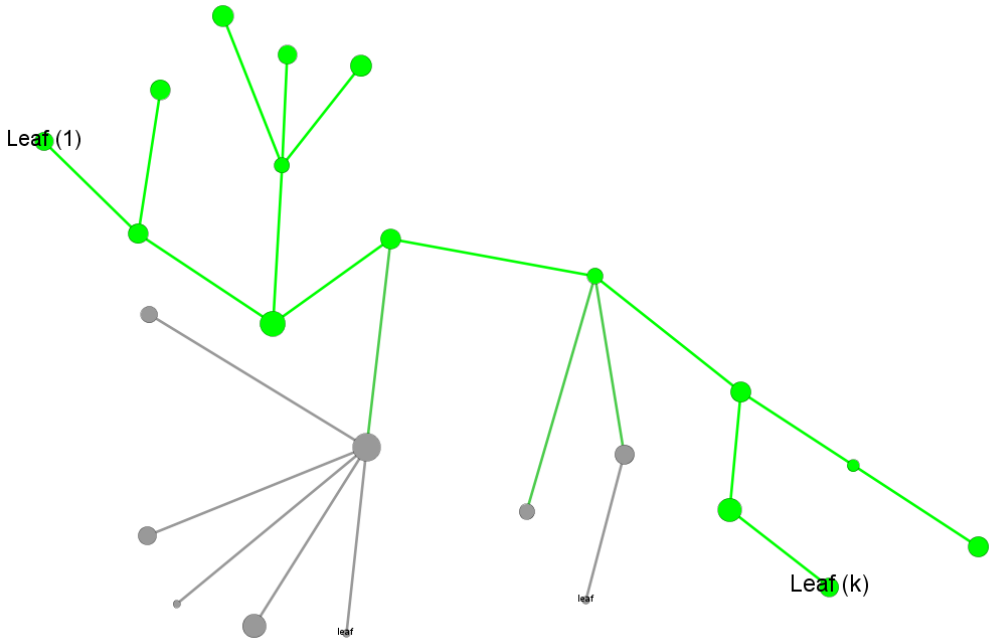


Figure 3.16 - Tree-based Network Topology

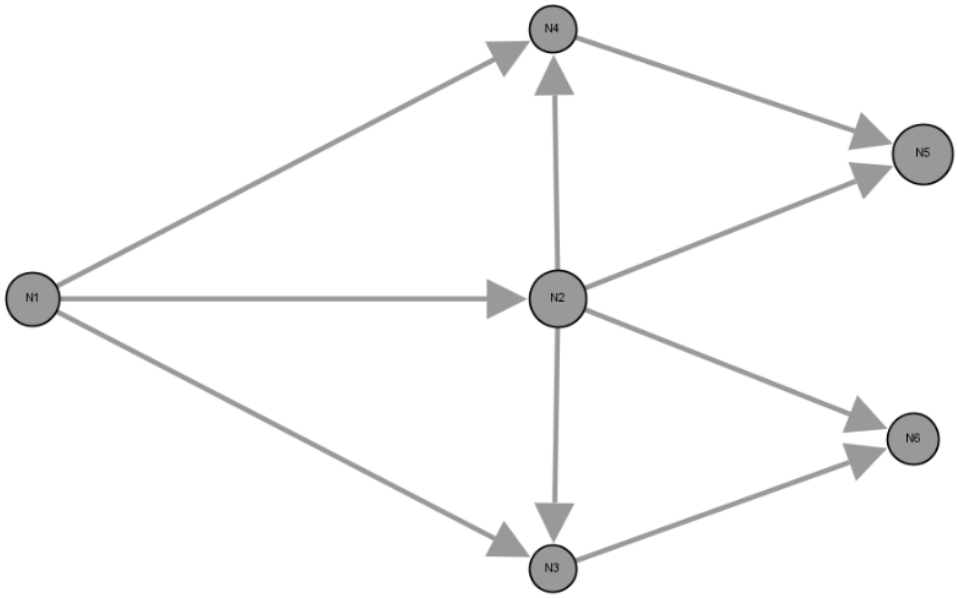
3.5.1 Network graph model

It is useful to define the term “graph” to assist network analyzing. In modern graph theory, a graph [92] can be classified by different types of edges into two main kinds, a directed graph, or digraph, and an undirected graph. These two types of graphs are mainly distinguished by the edge property: the directed graph has oriented edges while the undirected graph has no oriented edges. The other types of graphs that are all variants of the above two types of graphs are “mixed graph”, “multigraph”, “simple graph”, and “weighted graph”.

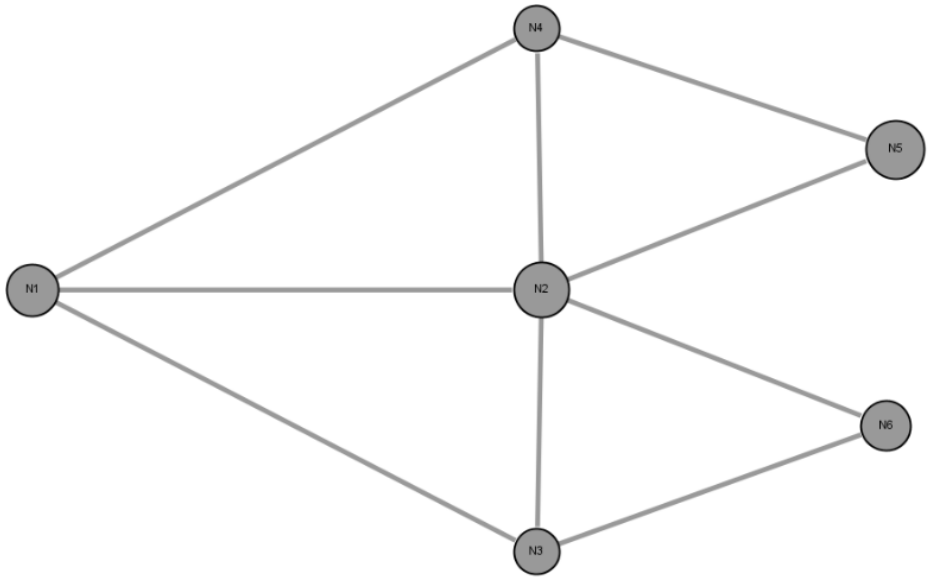
An undirected graph is a graph in which the edges have no orientation. If there is a graph G = (V, E), where V = {a, b, c, d} and E = {(a, b), (c, d), (d, a)}, then the edge (a, b) is identical to the edge (b, a), i.e., they are not ordered pairs. In networking, the

links are bidirectional with both “upward” transmitting and “downward” transmitting functionality. The maximum number of edges in an undirected graph without a loop is $n(n-1)/2$.

However, in a directed graph, the distinction appears from Edge set where E is a set of ordered pairs of vertices. Presenting as an expression, (a, b) is a different edge from (b, a), i.e., (a, b) is considered to be directed from node “a” to node “b”. Often “a” is called the head and “b” is the tail of an arrow, and (b, a) is treated as the inverted arrow of (a, b). Shown in Figure 3.17-(a) is an example of a directed graph which radiates from N1 to the rest of nodes, and Figure 3.17-(b) is an undirected graph with no oriented vertices.



(a)



(b)

Figure 3.17 - Example of a Directed Network (a) and an Undirected Network (b)

3.5.2 Topological metrics

3.5.2.1 Average Nodal Degree ($\bar{k}$)

This is the coarsest connectivity feature of any topology. The degree of a node is the number of edges neighbored to that node while the average nodal degree represents average of the degrees over all nodes in the network [92]. For example, in Figure 3.18, the degree of node X is 6 while the average nodal degree is 2.5.

$$deg_{avg} = \frac{node\ count \times value}{total\ number\ of\ node} = \frac{2\times1+4\times2+1\times4+1\times6}{8} = 2.5 \qquad (3.7)$$

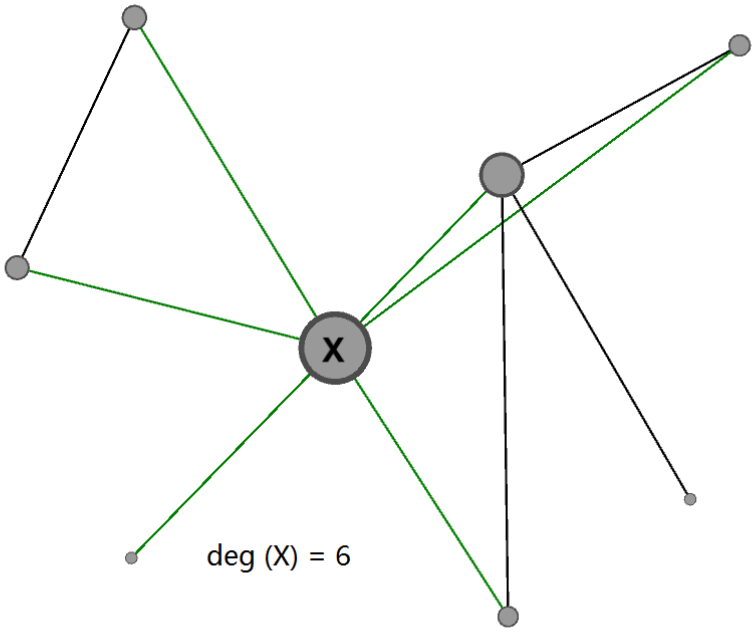


Figure 3.18 - Example of degree of a node X

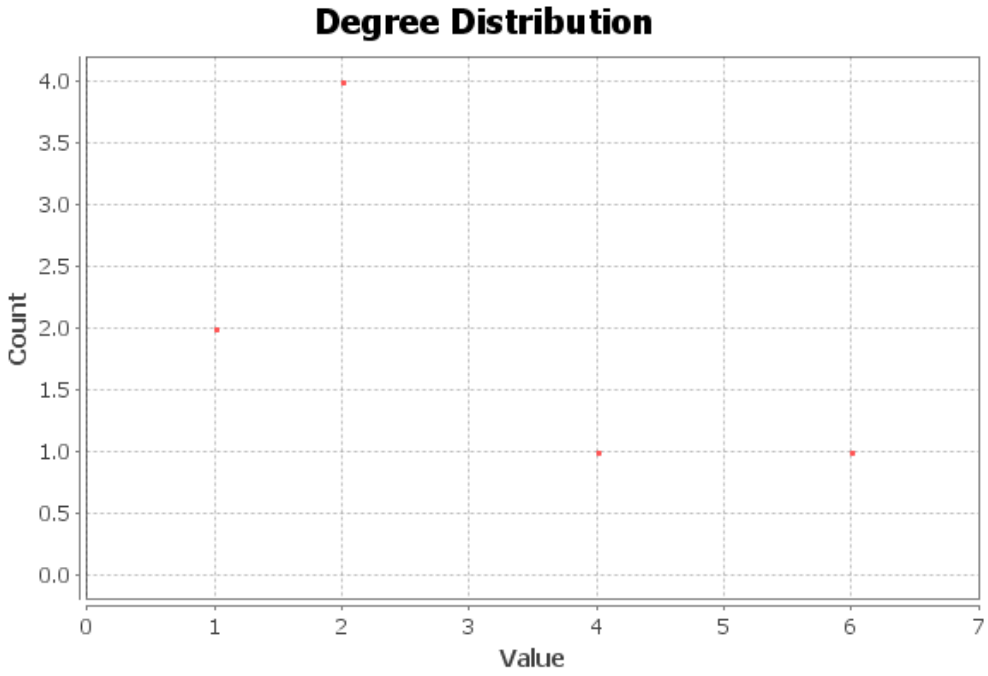
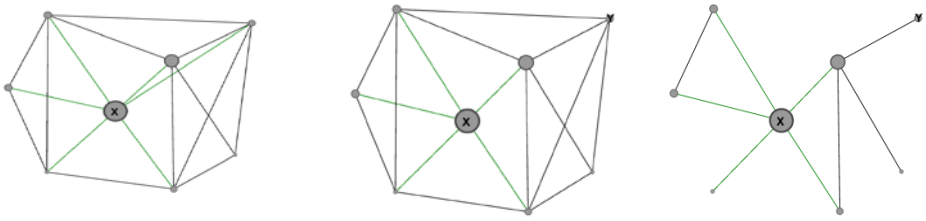


Figure 3.19 – Node degree distribution from 3.18 example

Networks with higher k are regarded as better-connected on average, and, consequently, are likely to be more robust. On one hand, “more robust” means that there are more chances to establish new connections such as the Figure 3.20, the average nodal degree increases to 4.5 when more connections are established based

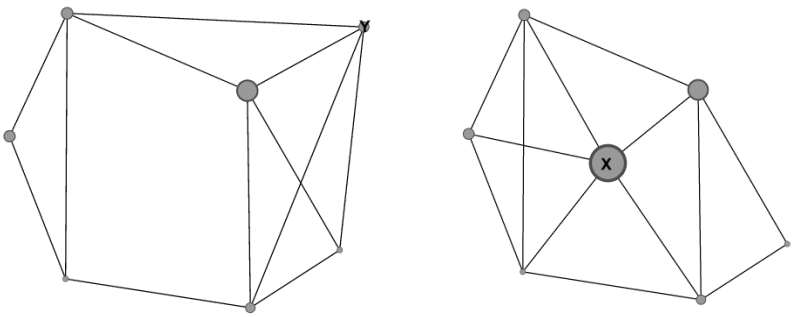
on Figure 3.20 (c). When one connection (X<->Y) is failed, the average nodal degree decreases to 4.25 while the same scenario from Figure 3.20 (c) to Figure 3.18 decreases the average degree from 2.5 to 2.25,



(a) (b) (c)

Figure 3.20 - (a) More robust model compared to Figure 3.19, (b) one connection fail compared to (a), (c) one connection fail compared to Figure 3.19.

However, if a node with a high nodal degree fails, potentially higher numbers of connections are also bound to be affected. For example, in Figure 3.21-(a), X obtains the highest nodal degree of 6, so the failure of node X makes the average nodal degree decrease from 4.5 to 3.429, while there are six connections that get disconnected. If node Y fails (Y with degree of 5) a little lower than the one with node X, the average nodal degree decreases from 4.5 to 3.714. From Table 3.2, the k of Failure 2 is higher than Failure 1, which means the failure of node X with a higher degree has more impacts on the system compared to the failure of node Y with a lower degree. So Y in Figure 3.21-(b) is more robustness than X in Figure 3.21-(a).



(a) (b)

Figure 3.21 - (a) Failure 1 - node X failure, (b) Failure 2 – node Y failure

Node Failure	Failure 1	Failure 2
Node selected	Node X	Node Y
K - Initial	4.5	4.5
K - After failure	3.429	3.714
Robustness	low	high

Table 3.2 - Robustness comparison according to node failure with different degree

However, in more complicated network architectures, this metric cannot indicate the overall network robustness because the network robustness is much more complex which has to be comprehensively evaluated. Thus, this metric by itself provides only a limited measure of the robustness of a network which is likely to vary depending on how the nodal degree is actually distributed over the graph.

3.5.2.2 Network Diameter

The diameter is, like the average nodal degree, another broad robustness metric of a network [94]. It is the longest of all the shortest paths between pairs of nodes. In general, one would wish the diameter of networks to be low. Scale-free networks generally have small diameters, but are not particularly robust in response to deliberate attacks, due to their relatively low value of node connectivity. Nonetheless, small-world networks represent a combination of the advantages of the properties of random networks (where no node is privileged by design) and scale-free networks (where there is a low diameter). We also note that expansion, the diameter of a network normalized by its size, could be also used in order to carry out a comparison analysis [94]. The length $max_{u,v}d(u,v)$ of the "longest shortest path" (i.e., the longest graph geodesic) between any two graph vertices (u,v) of a graph, where $d(u,v)$ is graph distance.

$$D = max_{u,v}d(u,v) \tag{3.8}$$

3.5.2.3 Average Shortest Path Length

The average shortest path length (ASPL) is calculated as an average of all the shortest paths between all the possible origin-destination node pairs of the network [95]. Generally, networks with smaller ASPL are more robust in network latency performing, but prone to lose connections due to less linking cardinal numbers. Therefore, in order to comprehensively compare network performance, ASPL cannot tell all, which is usually network robustness metric that with related to the network latency. d_{ij} denotes the distance between the vertices v_i and v_j, N is the vertices number, and the average path length L of an un-weighted network can be calculated by the formula:

$$L = \frac{2}{N(N-1)} \sum_{i \geq j} d_{ij} \tag{3.9}$$

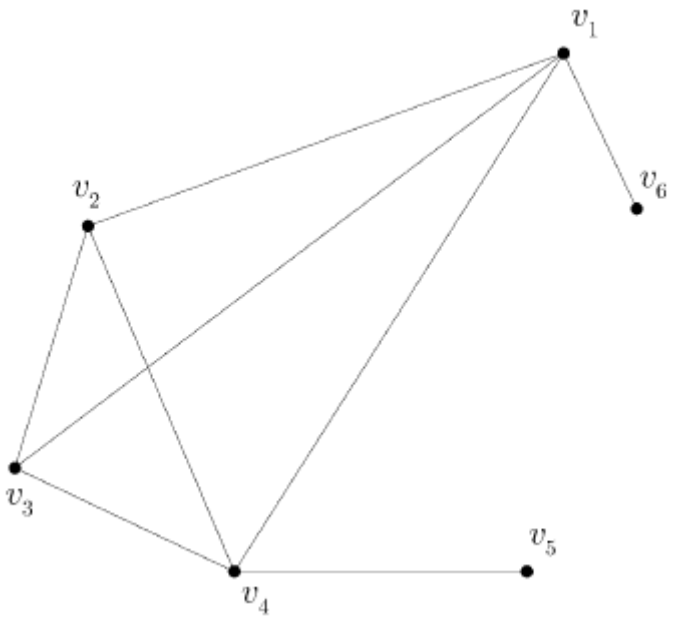


Figure 3.22 – ASPL: 6 nodes example

For example, $\sum_{i \geq j} d_{ij} = d_{12} + d_{13} + d_{14} + d_{15} + d_{16} + d_{23} + d_{24} + d_{25} + d_{26} + d_{34} + d_{35} + d_{36} + d_{45} + d_{46} + d_{56} = 1 + 1 + 1 + 2 + 1 + 1 + 1 + 2 + 2 + 1 + 2 + 2 + 1 + 2 + 3 = 23$, then $L = \frac{2}{6 \times 5} \times 23 = \frac{23}{15}$.

3.5.2.4 Betweenness Centrality

The betweenness centrality for a node in a network indicates the proportion of the node that lies on paths between other nodes in the network. A high proportion implies

an important node in a network which has a large influence on the transfer of messages through the network. The betweenness centrality of a node reflects the amount of control that this node exerts over the interactions of other nodes in the network [96].

The betweenness centrality of a node N is calculated as follows:

$$C_b(N) = \sum_{s \neq N \neq t} (\varphi_{st}(N)/\varphi_{st}) \quad (3.10)$$

Where *s* and *t* are nodes that different from *N*, φ_{st} refers to the number of shortest paths from node *s* to *t*, and $\varphi_{st}(N)$ denotes the number of shortest paths from *s* to *t* that *N* lies on.

For example, the betweenness centrality of node *b* is computed as follows:

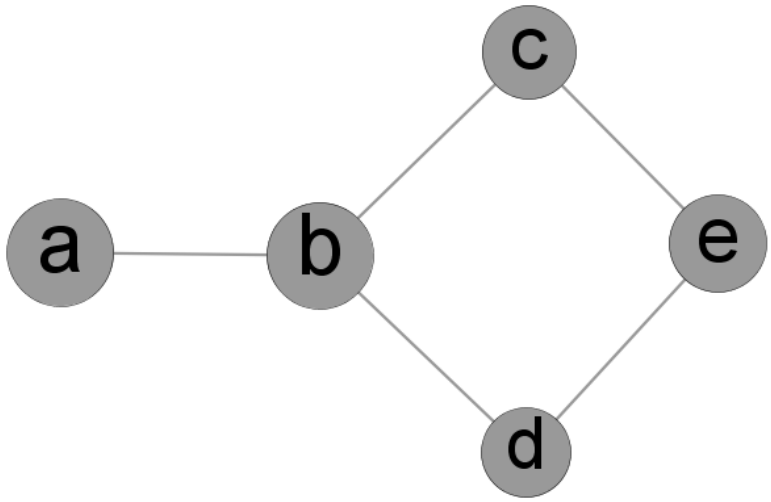


Figure 3.23 – Betweenness centrality: 5 nodes example

$$C_b(b) = \frac{\varphi_{ac}(b)}{\varphi_{ac}} + \frac{\varphi_{ad}(b)}{\varphi_{ad}} + \frac{\varphi_{ae}(b)}{\varphi_{ae}} + \frac{\varphi_{cd}(b)}{\varphi_{cd}} + \frac{\varphi_{ce}(b)}{\varphi_{ce}} + \frac{\varphi_{de}(b)}{\varphi_{de}} = 1/1 + 1/1 + 2/2 + 1/2 + 0 + 0 = 3.5 \quad (3.11)$$

3.5.2.5 Closeness Centrality

Closeness centrality is a measure of how fast information spreads from a given node to other reachable nodes in the network [97]. Nodes with high closeness centrality are important because they can reach the whole network more quickly than the other nodes. The node`s closeness centrality is measured by the reciprocal of its

average distance. The average distance of node v_i to other nodes is calculated as follows:

$$D_{avg}(v_i) = \frac{1}{n-1}\sum_{j\neq i}^{n} g(v_i, v_j) \quad (3.12)$$

, where n is the number of nodes, $g(v_i, v_j)$ is the length from node v_i to node v_j.

The closeness centrality of node v_i is measured as follows:

$$C_C(v_i) = \left[\frac{1}{n-1}\sum_{j\neq i}^{n} g(v_i, v_j)\right]^{-1} \quad (3.13)$$

For example, the closeness centrality of node b in Figure 3.23 is computed as follows:

$$C_C(b) = 1/\left(\left(L(b,a) + L(b,c) + L(b,d) + L(b,e)\right)/4\right)$$
$$= 4/(1 + 1 + 1 + 2) = 4/5 = 0.8$$

3.5.2.6 Eccentricity

In graph theory, the eccentricity $\mathrm{E}(v)$ of a vertex v is the greatest geodesic distance from v to another vertex. In a network, the eccentricity is regarded as the distance from a given starting node to the farthest node from it [98].

3.5.2.7 Eigenvector Centrality

Eigenvector centrality measures the influence of a node in a network. Each node is assigned a score on the basis of the concept that connections to high-scored nodes make more contributions to the score of the node than identical connections to low-scored nodes. PageRank by Google is a variant of eigenvector centrality measurement [99].

Eigenvector centrality can be measured by using adjacency matrix. For a given graph $\mathrm{G} = (V, E)$ with a set of vertex V and a set of edges E, and let $\mathrm{A} = (a_{v,t})$ be the adjacency matrix, if the vertex v is connected to vertex t, then the score of vertex v is defined as:

$$s_v = \frac{1}{\mu}\sum_{t\in M(v)} s_t = \frac{1}{\mu}\sum_{t\in G} a_{v,t}s_t \quad (3.14)$$

where $M(v)$ is a set of neighbors of v and μ is a constant value (referred as eigenvalue). Generally, only the greatest eigenvalue μ can result in the desired centrality measure [100].

3.5.3 Summary

Metrics such as average nodal degree, average weighted degree, and network diameter and radius, are necessary parameters for evaluating the robustness of a network. A larger average nodal and weighted degree, and a shorter diameter and radius tend to represent a robust network. However, the centrality metrics are important metrics that have great influence on a network.

Metric	**Summary**	**Metric Feature**
Average Nodal Degree	Measures the number of edges neighbored to that node	Coarse metric
Network Diameter	Measures the longest of all the shortest paths between pairs of nodes	Coarse Metric
Average Shortest Path Length	An average of all the shortest paths between all the possible origin-destination node pairs of the network	Shorter, less latency
Betweenness Centrality	Measures the proportion that lies on paths between other nodes in the network	Large influence
Closeness Centrality	Measures how fast information spreads from a given node to other reachable nodes in the network	Large influence
Eccentricity	Measures the distance from a given starting node to the farthest node from it	Medium influence
Eigenvector Centrality	Measures the influence of a node in a network	Large influence

Table 3.3 – Topological metric summary

Chapter 4 Simulation Studies

For the case study, the researcher firstly built an energy-aware DCN by conducting simulation studies in order to observe the impact of the underlying network connectivity on the DCNs` performance according to the fault tolerant perspective. A distributed DCN was then set up so as to evaluate different DCNs` performance under the failure conditions based on the solution of the first study. Both studies were conducted using a Network Analysis Tool (Gephi 0.8.2) and a Network Simulation Tool (CloudNetSim++ version 1.0 based on OMNeT++ Version 4.1 and Inet network framework).

4.1 Case studies

The structure of case studies in this chapter is shown in Figure 4.1 and 4.2 as following.

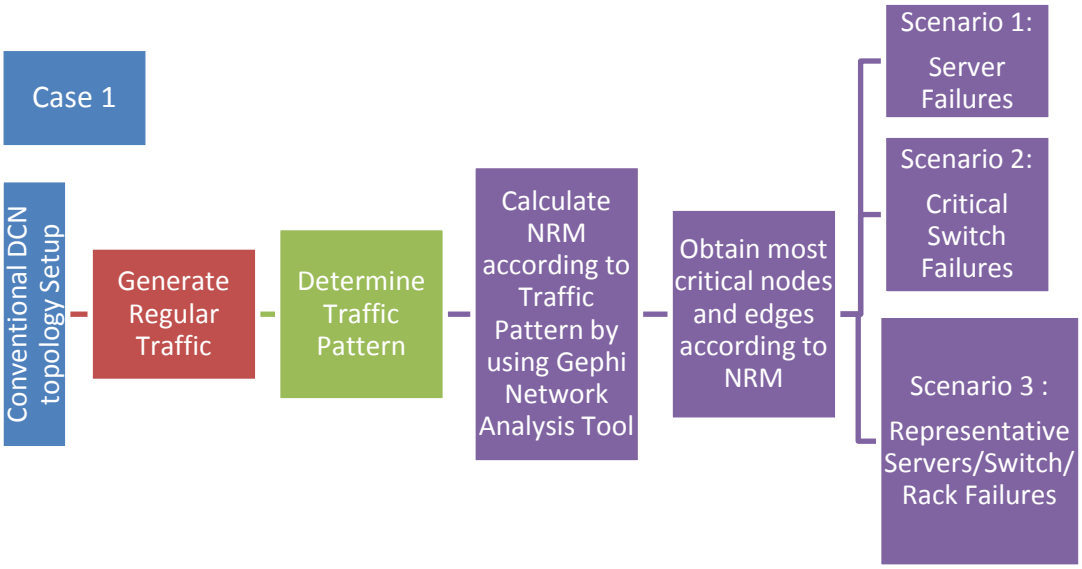


Figure 4.1 – Case 1 architecture

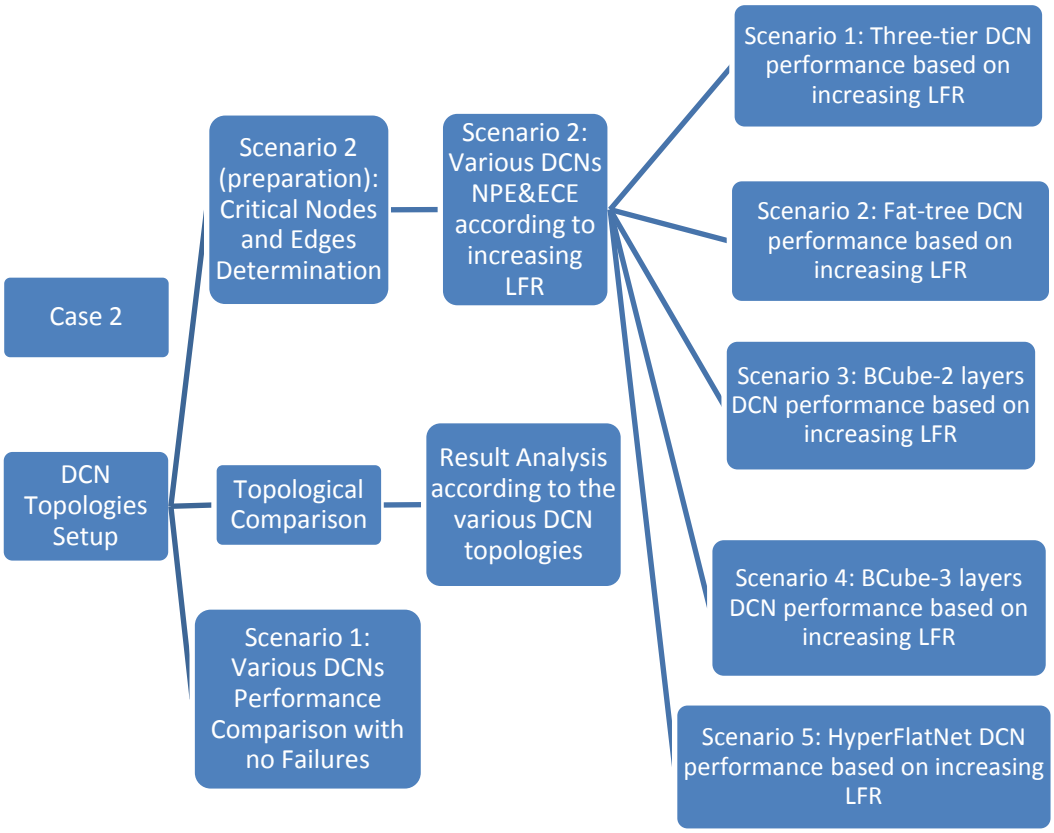


Figure 4.2 – Case 2 architecture

Notation	Representation
NRM	Network Robustness Metric
NP	DC Network Performance
NPE	DC Network Performance Evaluation
ECE	Energy Consumption Evaluation
LFR	Link Failure Ratio

Table 4.1 - Notice on notation representation of the use in simulation studies structure

As can be seen in Figures 4.1 and 4.2, the simulation study was divided into two processes. In the first process, robustness metrics were evaluated based on the network established, according to the analysis for each metric, and important nodes and edges were determined in this process according to NRM values. Various types of component failure based on important nodes and edges were simulated in CloudNetSim++ simulator as depicted in Scenarios 1, 2 and 3. The NP was analyzed and verified on the basis of node and link selection.

Initially, a representative conventional balanced-DCN with unbalanced edge weight was constructed in Gephi; the reason for this was to find important nodes and weights in the network by using network robustness metrics measuring technique so that the network performance simulation part could be treated as the verification of the results gained from Gephi. Moreover, the network QoS evaluation part is the first study was represented as the network performance and energy consumption versus the node failure scenarios so as to be able to compare between the different nodes.

The conclusion from the first study was a stepping stone to the second study as the second study generated more complicated realistic scenarios by comparing different trendy DCNs to achieve the main goal of this thesis. Initially, five DCN architectures were set up, and the architectures were comprehensively compared in topological view. Then in Scenario 1, network performance of various DCNs was recorded and compared using CloudNetSim++. In Scenario 2, as a preparation process, the critical nodes and edges were first determined by comparing network robustness metrics and centrality metrics in Gephi. The main part of Scenario 2 was gaining NPE[1] and ECE for various DCNs according to increasing LFR.

[1] NPE includes average network throughput, average packet delay, packet drop ratio, and the total number of packets received.

4.2 Case 1: Network Performance Evaluation (NPE) according to Network Robustness Metrics (NRM)

4.2.1 Topology Setup

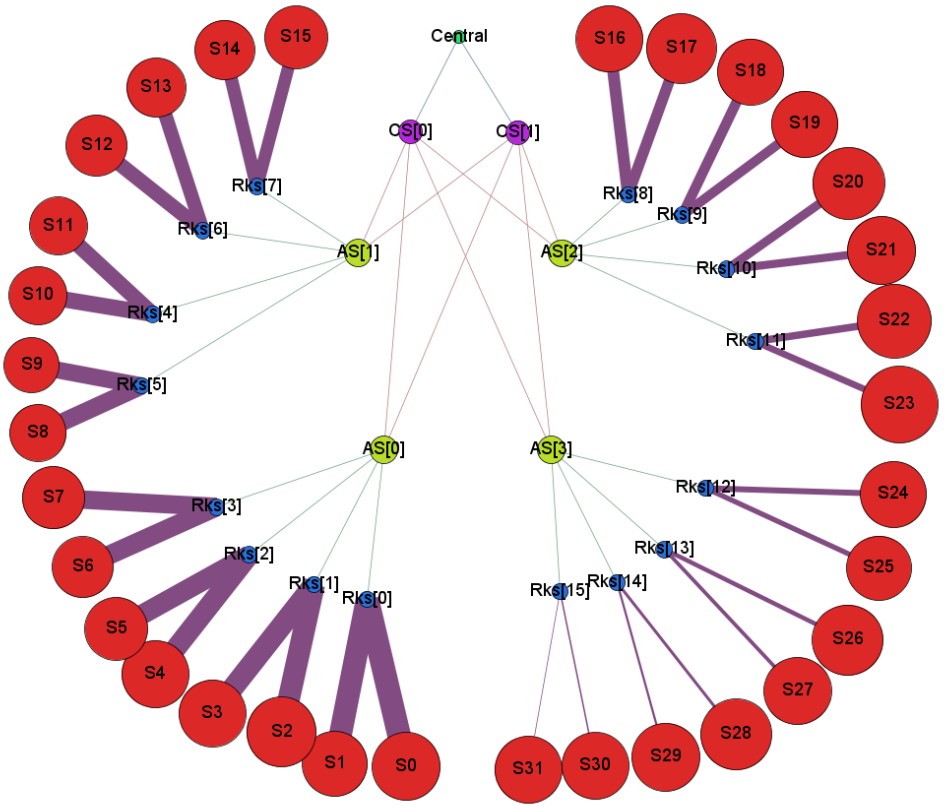


Figure 4.3 – Case 1 network setup

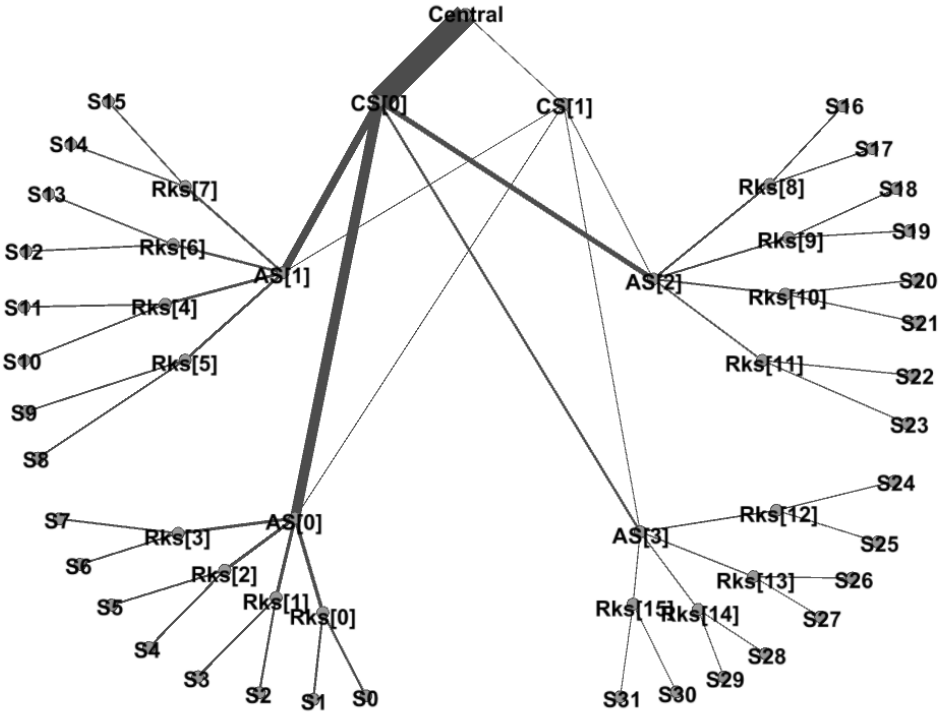


Figure 4.4 – Case 1 network implementation in Gephi

*The edge thickness is depicted according to the edge weight size, while the weight in this case represents the traffic flow.

Notation	**Representation**
Central	Central Switch which connects outside DC network
CS[n]	Core Switch [n]
AS[n]	Aggregation Switch [n]
Rks[n]	Edge Router [n], but shown as Racks[n] in the figure
Sn	*n*th server
*****	implies all (i.e. CS[*] means all core switches in the network)

Table 4.2 - Notice on notation representation of node used in this network

4.2.2 Node Eigenvector Centrality Evaluation

Eigenvector Centrality	Value (in order of importance)
CS[*]	1
AS[*]	0.821
Central	0.514
Edge[*] (display as Rks[*])	0.268
Server[*]	0.077

Table 4.3 – Case 1 network: Eigenvector Centrality evaluation

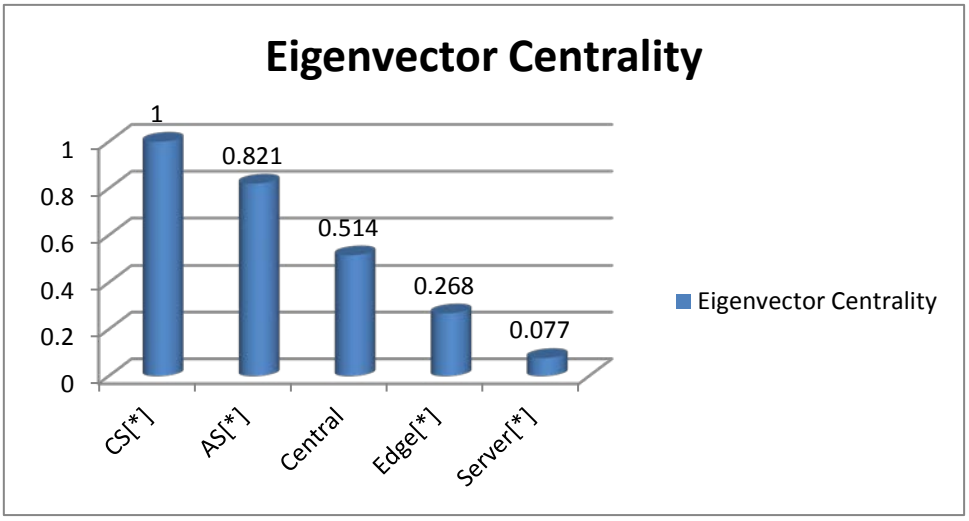


Figure 4.5 – Case 1 network: Eigenvector Centrality ranking

Core switches and aggregation switches own the highest eigenvector centrality which indicates that they are the most important nodes in such network, with important neighbors connected as well.

4.2.3 Node Betweenness Centrality Evaluation

Betweenness Centrality	Value (in order of frequency)
AS[*]	558.2
CS[*]	533
Edge[*] (display as Rks[*])	105
Central	0.2
Server[*]	0

Table 4.4 – Case 1 network: Betweenness Centrality ranking

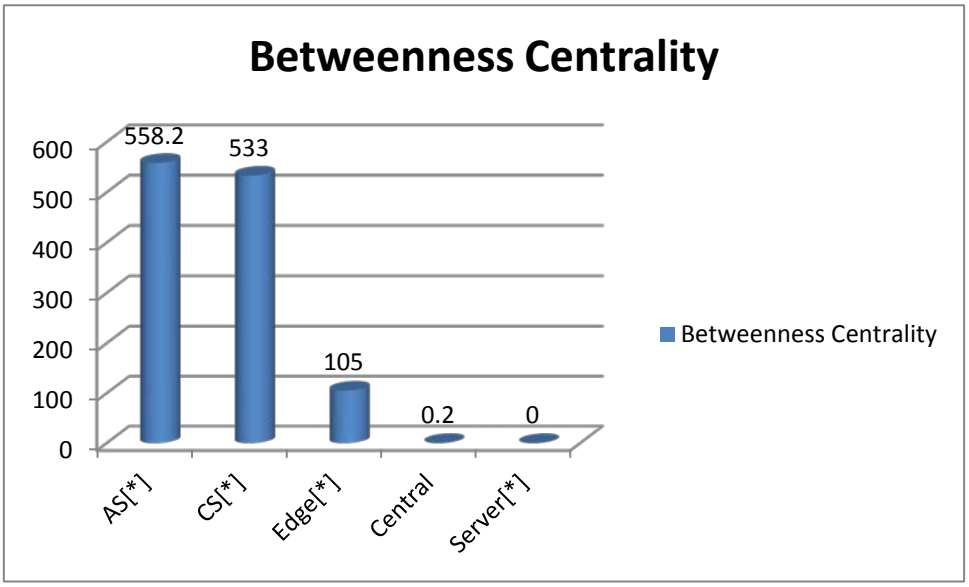


Figure 4.6 - Case 1 network: Betweenness Centrality ranking

According to the rank, aggregation switches which lie on a high proportion of paths between other nodes in the network are the most important nodes, which is to say that aggregation switches held the greatest responsibility in this study. Similarly, core switches also hold an important position for relaying packets to farther destinations. On the other hand, each edge router had 1 in 5 chances of being walked through that packet than each aggregation switch.

4.2.4 Node Closeness Centrality Evaluation

Closeness Centrality	Value (in order of distance)
CS[*]	0.402
AS[*]	0.353
Central	0.293
Edge[*] (display as Rks[*])	0.27
Server[*]	0.214

Table 4.5 - Case 1 network: Eigenvector Centrality ranking

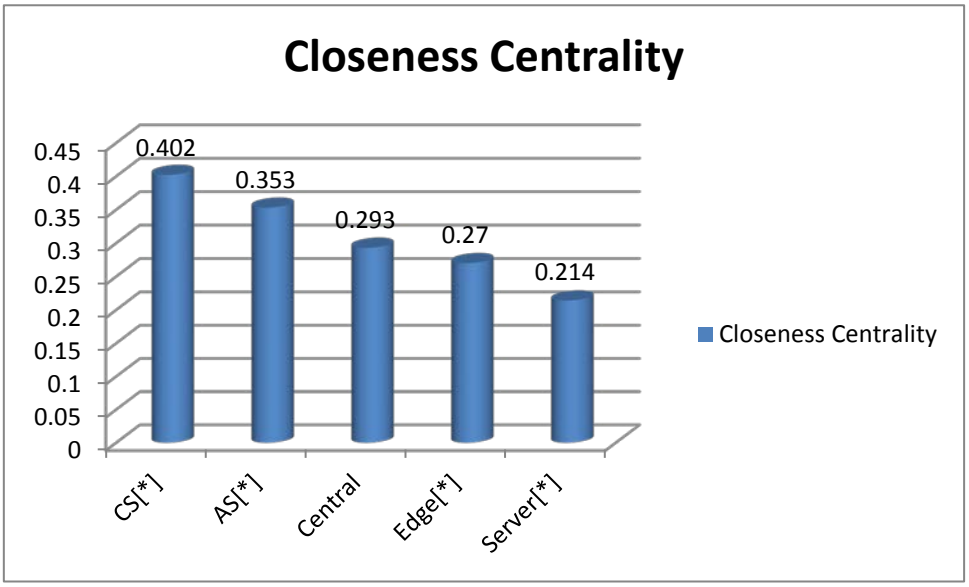


Figure 4.7 - Case 1 network: Closeness Centrality ranking

The core switches and aggregation switches obtained the largest value of closeness centrality which means that these two types of switches could quickly communicate with other nodes in the network. The less the average distance to other nodes, the more efficiently it can relay the message, especially in a role of a switch. The results showed that the core layer switches and aggregation layer switches gain the ability of fast communication with other nodes in the network.

4.2.5 Node Eccentricity Evaluation

Eccentricity	**Value (in order of distance)**
CS[*]	3
AS[*]	4
Central	4
Edge[*] (display as Rks[*])	5
Server[*]	6

Table 4.6 - Case 1 network: Eccentricity Centrality ranking

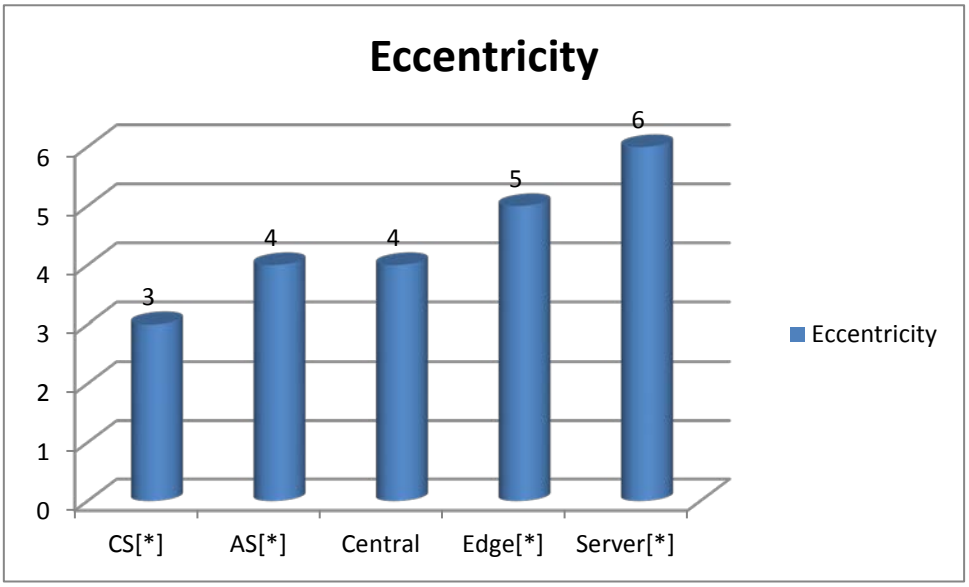


Figure 4.8 - Case 1 network: Eccentricity Centrality ranking

As expected, servers had the largest value of eccentricity. They had distance of six hops from one server to the farthest server. For example, the server connected by AS[0] had to go through six hops to the server connected by AS[1], but four hops to a neighboring rack which was connected by the same aggregation switch, and two hops to the server inside a rack. On the other side, core switches and aggregation switches had the lowest eccentricity value which means they were lying on the "center" of the network.

4.2.6 Node Degree Evaluation

Nodal Degree	**Value (in order of size)**
AS[*]	6
CS[*]	5
Edge[*] (display as Rks[*])	3
Central	2
Server[*]	1

Table 4.7 - Case 1 network: Node degree ranking

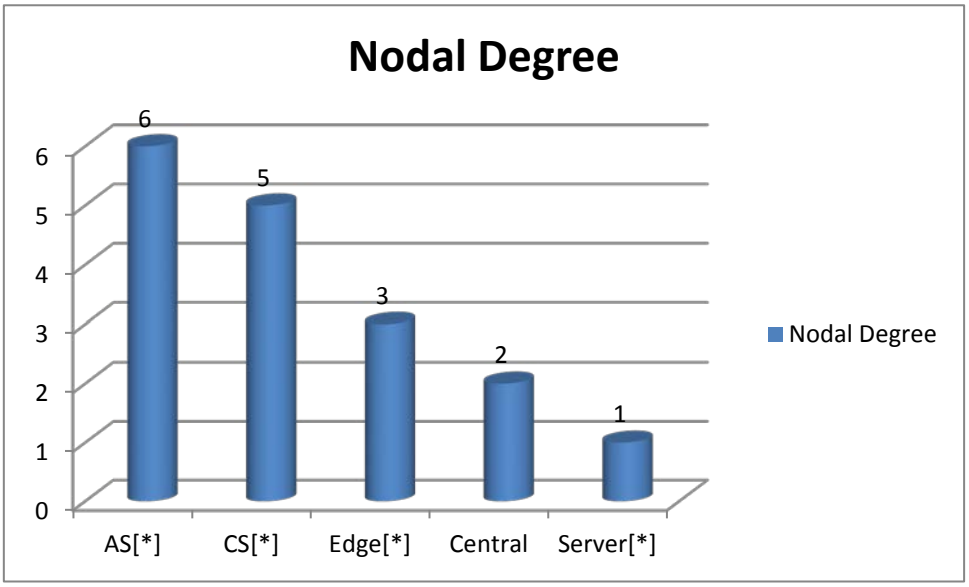


Figure 4.9 - Case 1 network: Nodal degree ranking

Aggregation switches achieved highest nodal degree of 6 because each aggregation switch has 6 ways out to its neighbors.

4.2.7 Node Weighted Degree Evaluation

Nodal Weighted Degree	Value (in order of size)
CS[0]	1056
Central	529
AS[0]	457
AS[1]	329
AS[2]	201
Edge[0] (display as Rks[0])	126
Edge[1] (display as Rks[1])	118
Edge[2] (display as Rks[2])	110
Edge[3] (display as Rks[3])	102
Edge[4] (display as Rks[4])	90
Edge[5] (display as Rks[5])	82
Edge[6] (display as Rks[6])	78
AS[3]	73
Edge[7] (display as Rks[7])	70

…continued	
Edge[8] (display as Rks[8])	62
Edge[9] (display as Rks[9])	54
Edge[10] (display as Rks[10])	46
Edge[11] (display as Rks[11])	38
Server[0]	32
Server[1]	31
Server[2]	30
Edge[12] (display as Rks[12])	30
Server[3]	29
Server[4]	28
Server[5]	27
Server[6]	26
Server[7]	25
Server[8]	24
Server[9]	23
Server[10]	22
Server[11]	21
Server[12]	20
Server[13]	19
Server[14]	18
Server[15]	17
…	…
Server[31]	1

Table 4.8 - Case 1 network: Weighted degree ranking

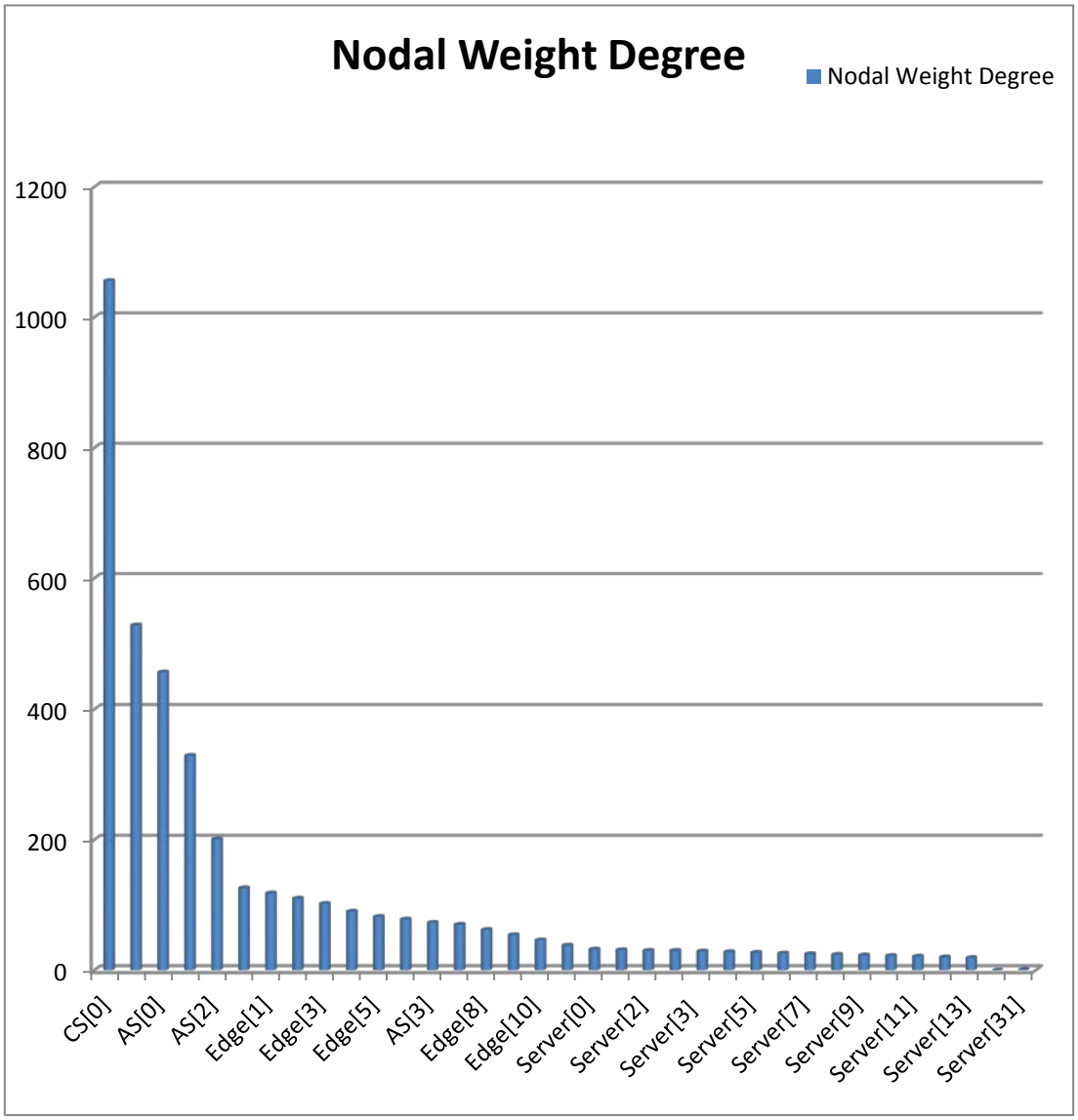


Figure 4.10 - Case 1 network: Nodal weighted degree ranking

4.2.8 Edge Weighted Degree Evaluation

Edge Weighted Degree	Value (in order of size)
Central – CS[0]	528
CS[0] – AS[0]	228
CS[0] – AS[1]	164
CS[0] – AS[2]	100
AS[0] – Edge[0]	63
AS[0] – Edge[1]	59
AS[0] – Edge[2]	55

…continued	
AS[0] – Edge[3]	51
AS[1] – Edge[4]	47
AS[1] – Edge[5]	43
AS[1] – Edge[6]	39
CS[0] – AS[3]	36
AS[1] – Edge[7]	35
Edge[0] – Server[0]	32
Edge[0] – Server[1]	31
Edge[0] – Server[2]	30
Edge[0] – Server[3]	29
Edge[1] – Server[4]	28
Edge[1] – Server[5]	27
…	…
Edge[15] – Server[31]	1

Table 4.9 - Case 1 network: Edge weighted ranking

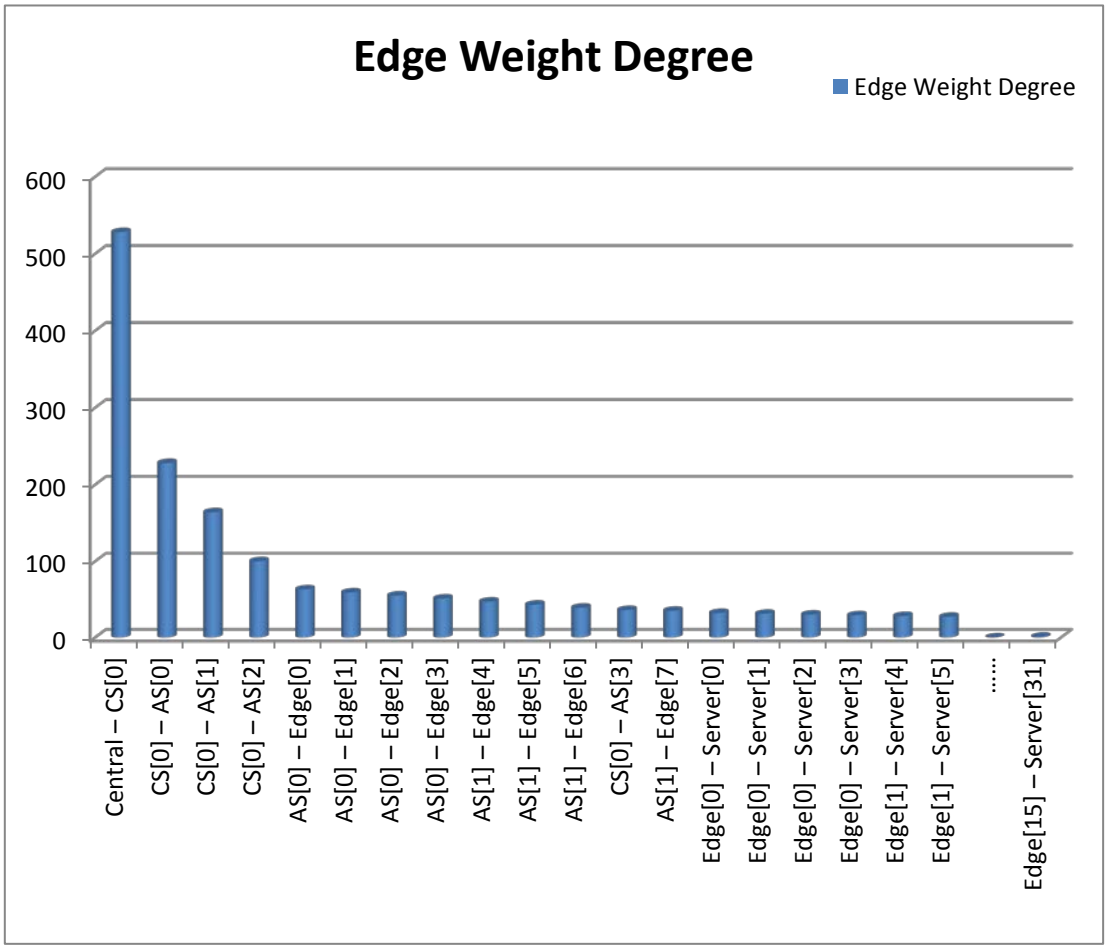


Figure 4.11 - Case 1 network: Edge weighted ranking

4.2.9 Critical Nodes determination

According to centrality metrics, CS and AS are the most important nodes in the network, and by the result shown in the weighted degree, AS[0] was determined as the most important switch in this network. Server[0] was regarded as the most important server, and Rack[0] was the most important rack. To verify this assumption, the simulation was conducted from latitudinal and longitudinal dimensions. Comparisons were carried out among servers; among aggregation switches; and between AS, Rack and server.

4.2.10 Network Performance Evaluation (NPE) Simulation setup

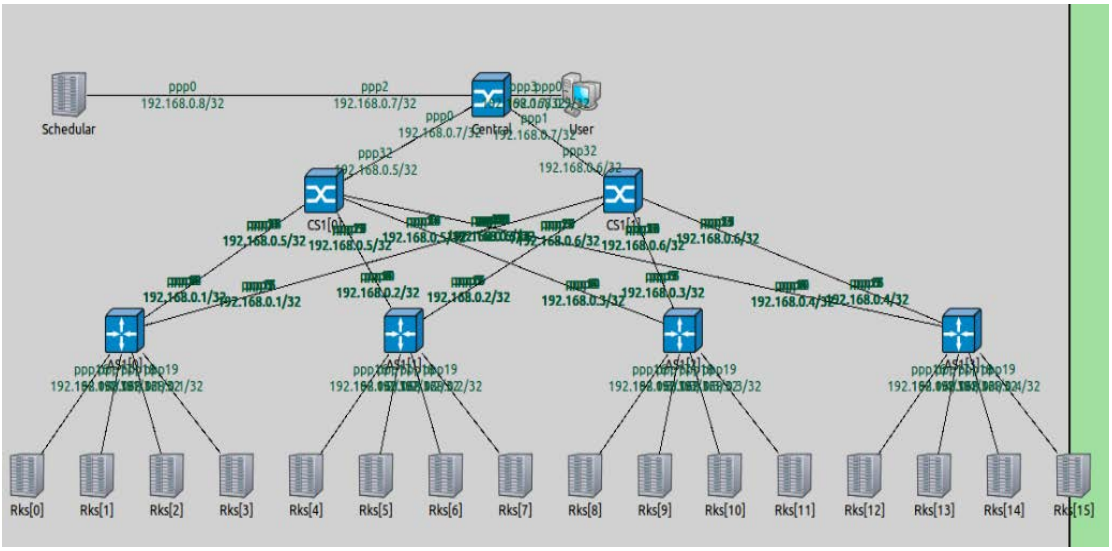


Figure 4.12 - Case 1: Simulation setup

Parameter	Value	Description
DCN	Three-Tier architecture	Conventional DCN
No. of Core Switch (CS)	2	Shown as CS[0] & CS[1]
No. of Aggregation Switch (AS)	4	Shown as AS[0]-AS[3]
No. of Rack (Rks)	16	Shown as Rack[0]-Rack[15]
No. of Server per Rack	2	Shown as Rack[*].server[0], Rack[*].server [1]
Protocol	UDP	-
Traffic Type	All-To-All	Server-To-Server
Server workload Distribution	Diverse	From high-load to low-load (Rks[0].server[0] to Rks[15].server[1])
Packet Size	1000 bytes	-
Send Interval	5 s	-
Queue Type	Drop Tail Queue	Queuing Mechanism

…continued		
Queue Capacity	1000	-
Power Management	DVFS & DPM	Dynamic Voltage/Frequency Scaling & Dynamic Power Management (Fixed Timeout τ)
Simulation Time	2500 sim-seconds	Simulation Time

Table 4.10 – Simulation parameters setting

4.2.11 Traffic Generation

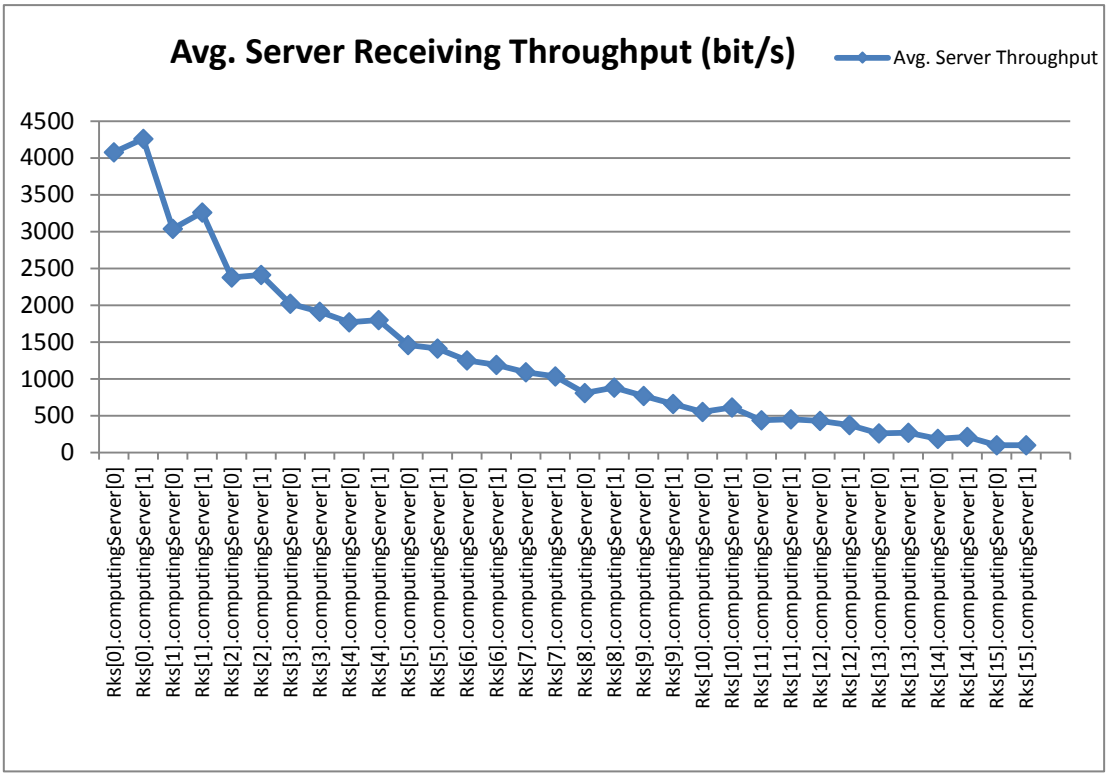


Figure 4.13 – Simulation server traffic generation

As realized in CloudNetSim++, the traffic generated in this case was set as unbalanced and skewed distribution, from the left to the right, and each server sent an equal number of packets (400) to other servers, while the servers located to the left, i.e., Rack[0].server[0], received the larger number of packets compared to the right, i.e., Rack[15].server[1], as shown in the figure above.

4.2.12 Case 1 - Scenario 1: Network Performance according to Server Failures

The sampling servers were selected based on the principle of equipartition in the range of server size, so from the left to the right were, in order, the server with the lowest throughput_received to the highest throughput_received.

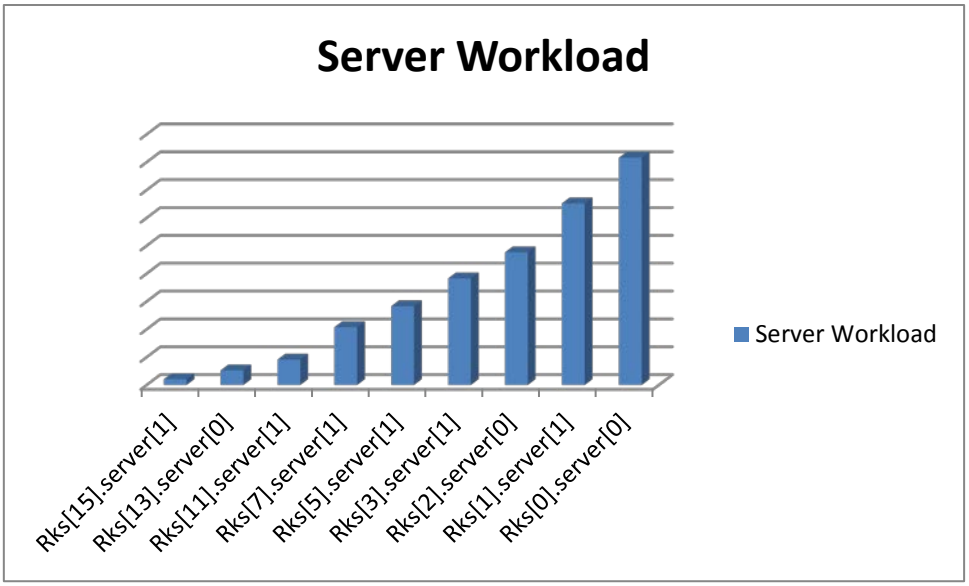


Figure 4.14 – Server workload distribution

Rack[n] selection	Server[n] selection	Avg. Throughput_rx (bit/s)
Rks[15]	Server[1]	103.0
Rks[13]	Server[0]	260.75
Rks[11]	Server[1]	453.9
Rks[7]	Server[1]	1036.57
Rks[5]	Server[1]	1413.21
Rks[3]	Server[1]	1912.18
Rks[2]	Server[0]	2378.96
Rks[1]	Server[1]	3257.79
Rks[0]	Server[0]	4078.67

Table 4.11 - Sample servers' description and servers` receiving throughput

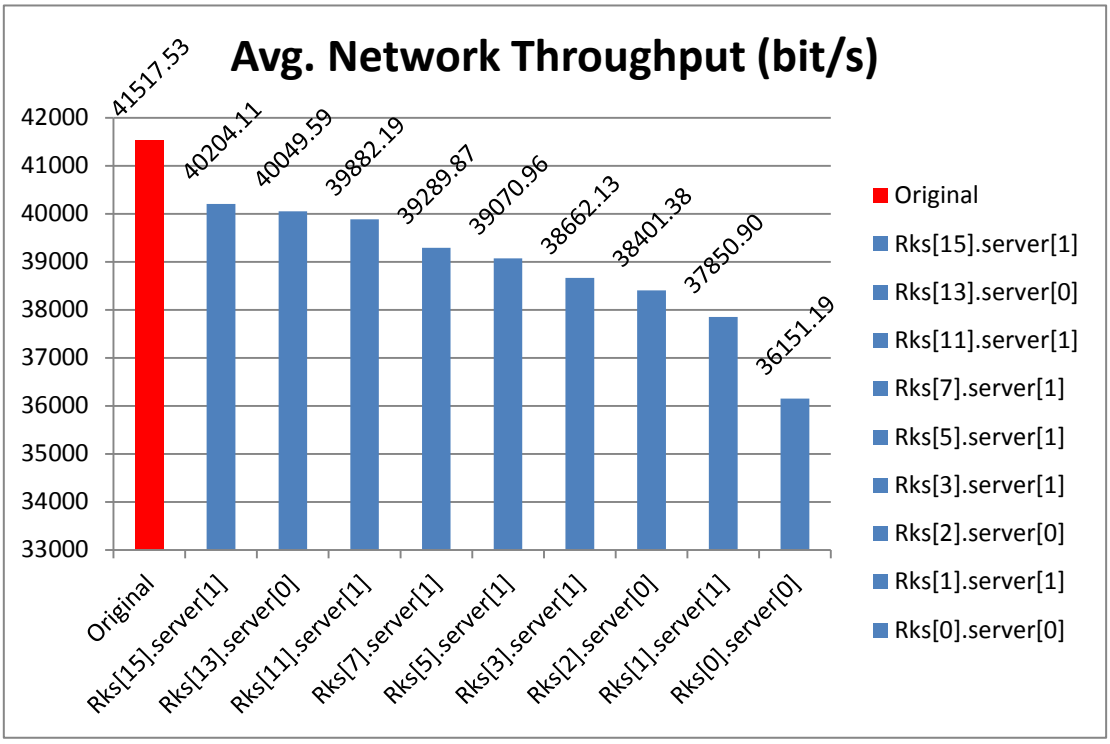


Figure 4.15 – Avg. Network Throughput (bit/s)

From Figure 4.15, it can be seen that the failure of the server with low throughput_rx remained high average network throughput, and vice versa. The failure on different servers generated various average network throughputs for packet receiving (throughput_rx), while the network with no failures achieved 41,500 bits/s throughput_rx, but the failure on higher-workload server caused a larger impact on the overall network performance. Originally, Rks[0].server[0] was the node that made up the highest throughput to the network which contributes 9.82% to the entire network, while Rks[15].server[1] contributed the smallest portion of the network throughput of only 0.248%. However, the node failure would further degrade the entire network performance due to the completed cloud tasks being reduced because of the decreasing total number of packets received. The failure on Rks[15].server[1] made average network throughput decrease by 3.1% while Rks[0].server[0] decreased 12.9% on the overall network average throughput.

	Rks[15].server[1]	Rks[0].server[0]
Contribution to overall avg. network throughput (%)	0.248%	9.82%
Avg. network throughput degradation after failure (%)	3.1%	12.9%

Table 4.12 – Representative node network throughput ratio and degradation after failure

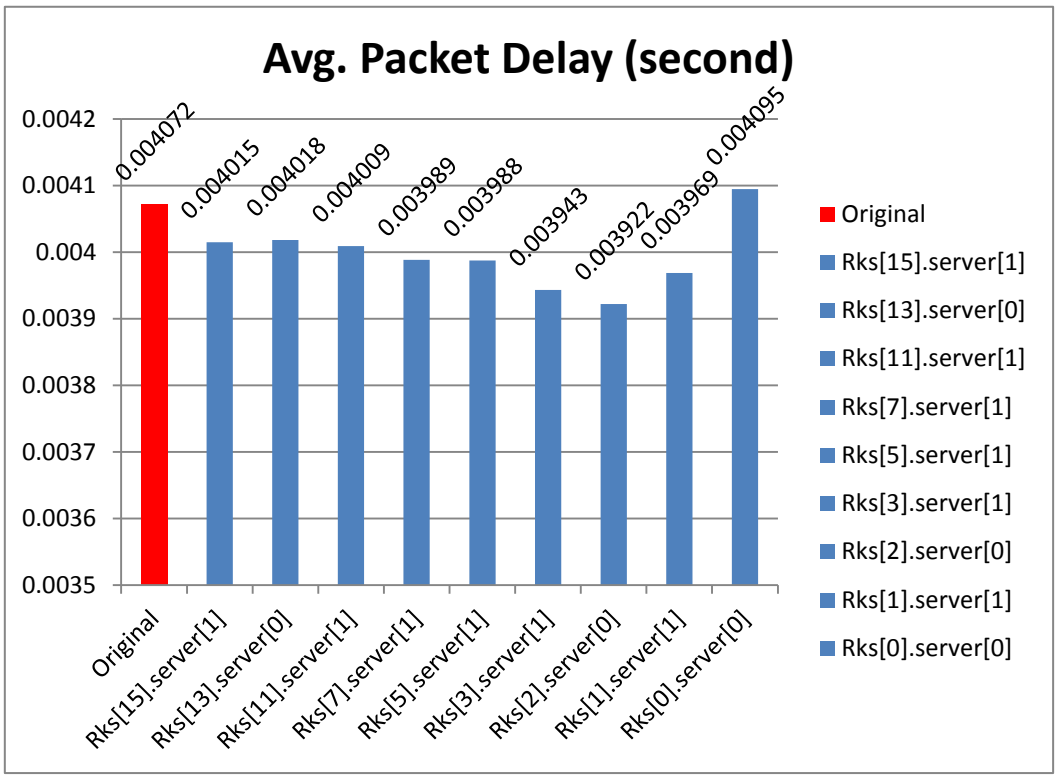


Figure 4.16 – Avg. packet delay (second)

The avg. packet delay did not vary too much on the basis of failures, which is still in the accepted range of 0.0039 - 0.0041 seconds.

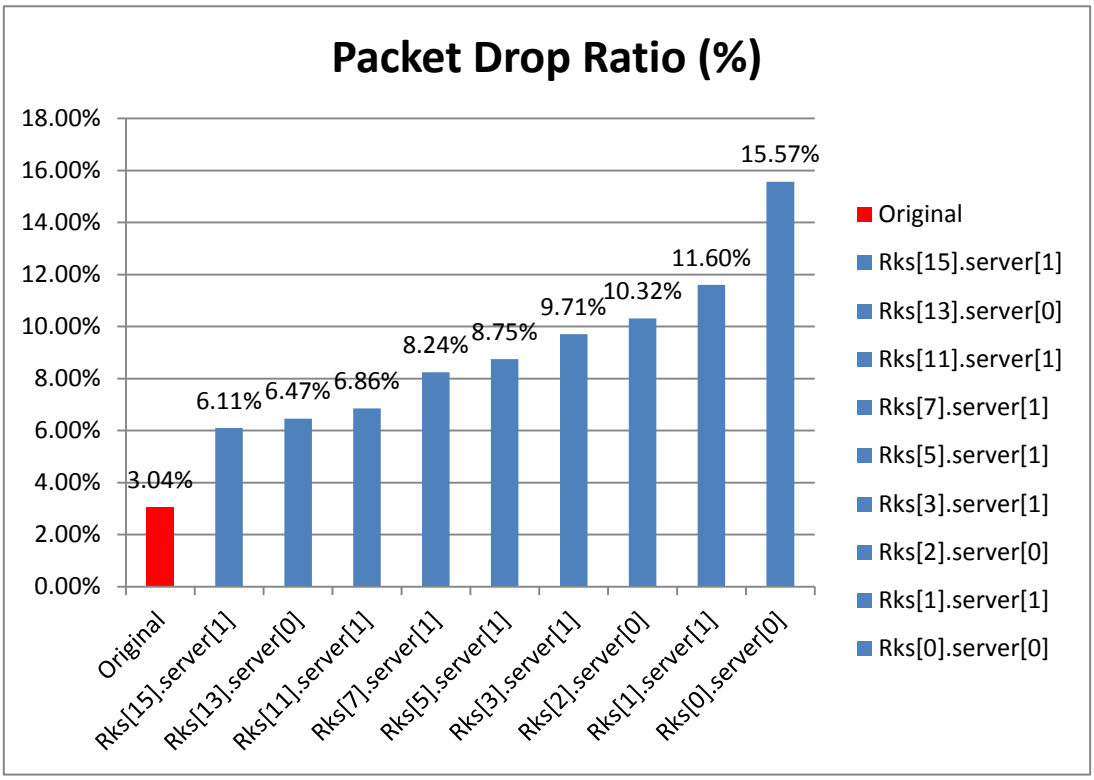


Figure 4.17 – Packet drop ratio (%)

The packet drop ratio result presented a reverse graph to the one for an average network throughput. Failures on high-loaded servers produced a higher packet drop ratio. The packets which were originally going to the failed server had to be dropped by the switches taking charge of that data flow. The failure on the most important node generated the highest packet drop ratio which was 15.57%, five times higher than the original network`s ratio.

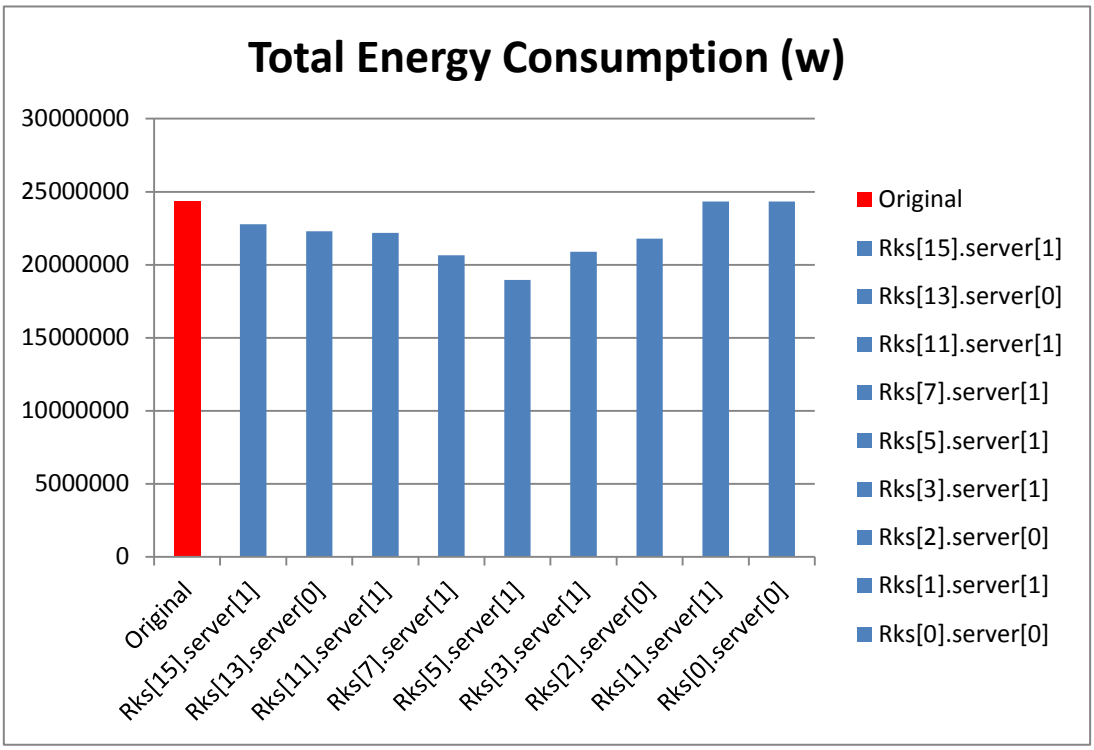


Figure 4.18 – Total energy consumption (w)

The energy consumption for the network was considered to be the sum of computing servers and the communicational components such as the switches. According to the GreenCloud [11], the power for supporting the operation of the switches accounted for a large portion (approximately 1/3 under normal operating conditions) of the total power consumption. The DVFS technique automatically decreased the energy consumption on the basis of CPU frequency adjustment, and servers with peak load correspond to the high CPU frequency. According to the DVFS model, the power consumption of a server was proportional to the CPU frequency, and the high-loaded server consumed more power rather than the low-loaded server. When failure occurred on a server, packets may be dropped by a switch in varying degrees, and transmitting to a failed server was blocked at the connected edge switch, which caused a high workload to the switch so that the energy consumption for the switch increased.

$$E_T = P_{switch} + P_{server} \tag{4.1}$$

, where E_T is the total network energy consumption, P_{switch} is the power supplied for switches and P_{server} represents the server power consumption. The failed server caused increment “a” on the switch power because of the increasing switch work load,

and reduction “b” on the server power, so that the Total power consumption after failure can be calculated as:

$$E_T` = (P_{switch} + a) + (P_{server} - b) \quad (4.2)$$

So that

$$E_T` = P_{switch} + P_{server} + (a - b) \quad (4.3)$$

, where $E_T`$ indicates the total energy consumption after server failure. While $|a| < |b|$ due to the reality of hardware component property, the power needed for supplying components of a server is always higher than the one for a switch, and conditions change in a server so it needs more power supplementing compared to the switch. Therefore, the absolute increment for switch power cannot be beyond the absolute reduction for server power. This is to say, a – b is always < 0, but various “a” and “b” makes the reduction of E_T in varying degrees. As Figure 4.18 shows, as the servers failed in order from low loaded to high loaded, the reduction of E_T became larger, however, the value of (a – b) became smaller, so that the reduction of E_T was not so obvious. However, despite the theoretical assumption, in an extreme condition, the $E_T`$ could be larger than E_T, which indicates that $|a| > |b|$, could, in this case, mean more power will be consumed for supplying the intensive workload for switches which could overwhelm the power reduction of the server, under various conditions of failure.

4.2.13 Case 1 - Scenario 2: Network Performance according to Aggregation Switch Failures

According to the critical node determination, AS[0] was the most influenced node in the network, therefore, scenarios of aggregation switch failures were simulated to have a comprehensive comparison among equal switches.

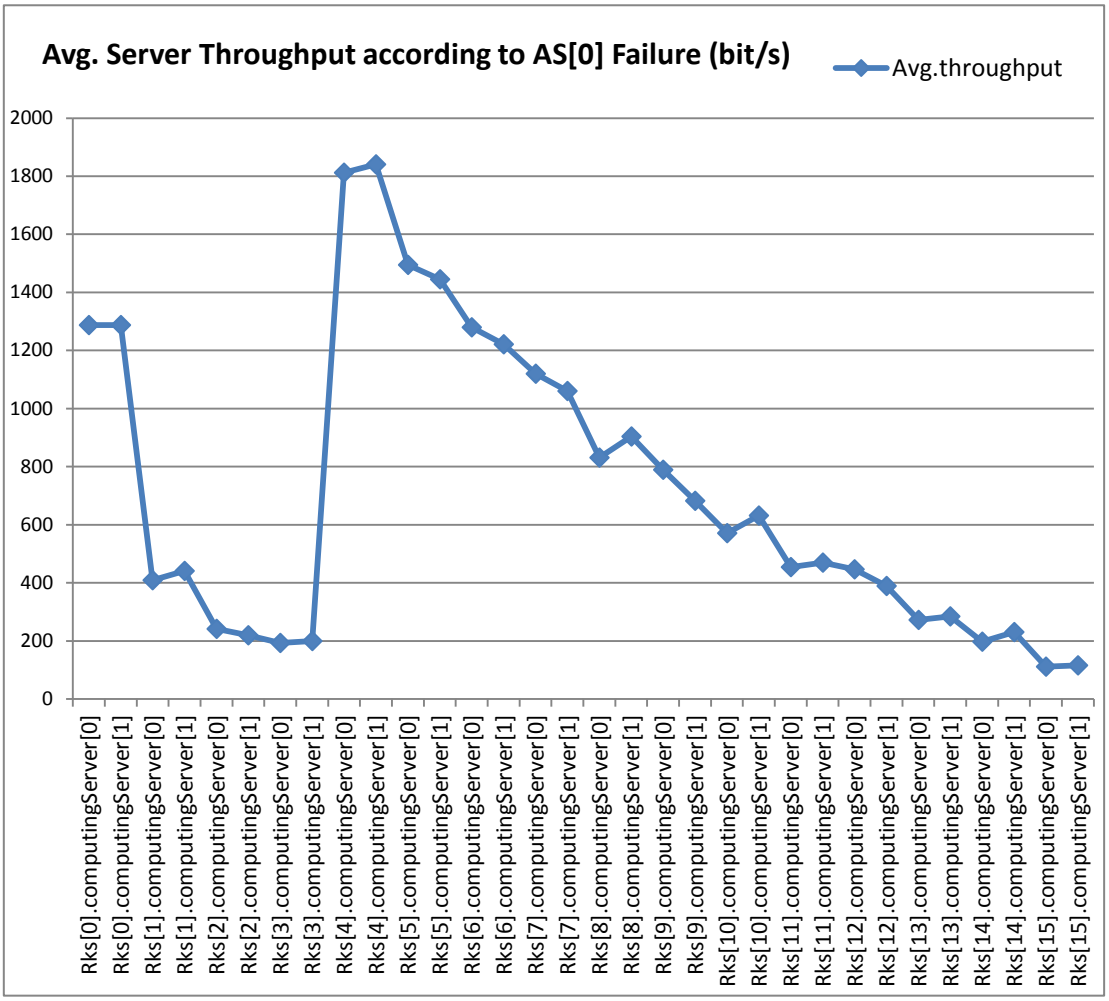


Figure 4.19 - Avg. server throughput according to AS[0] Failure (bit/s)

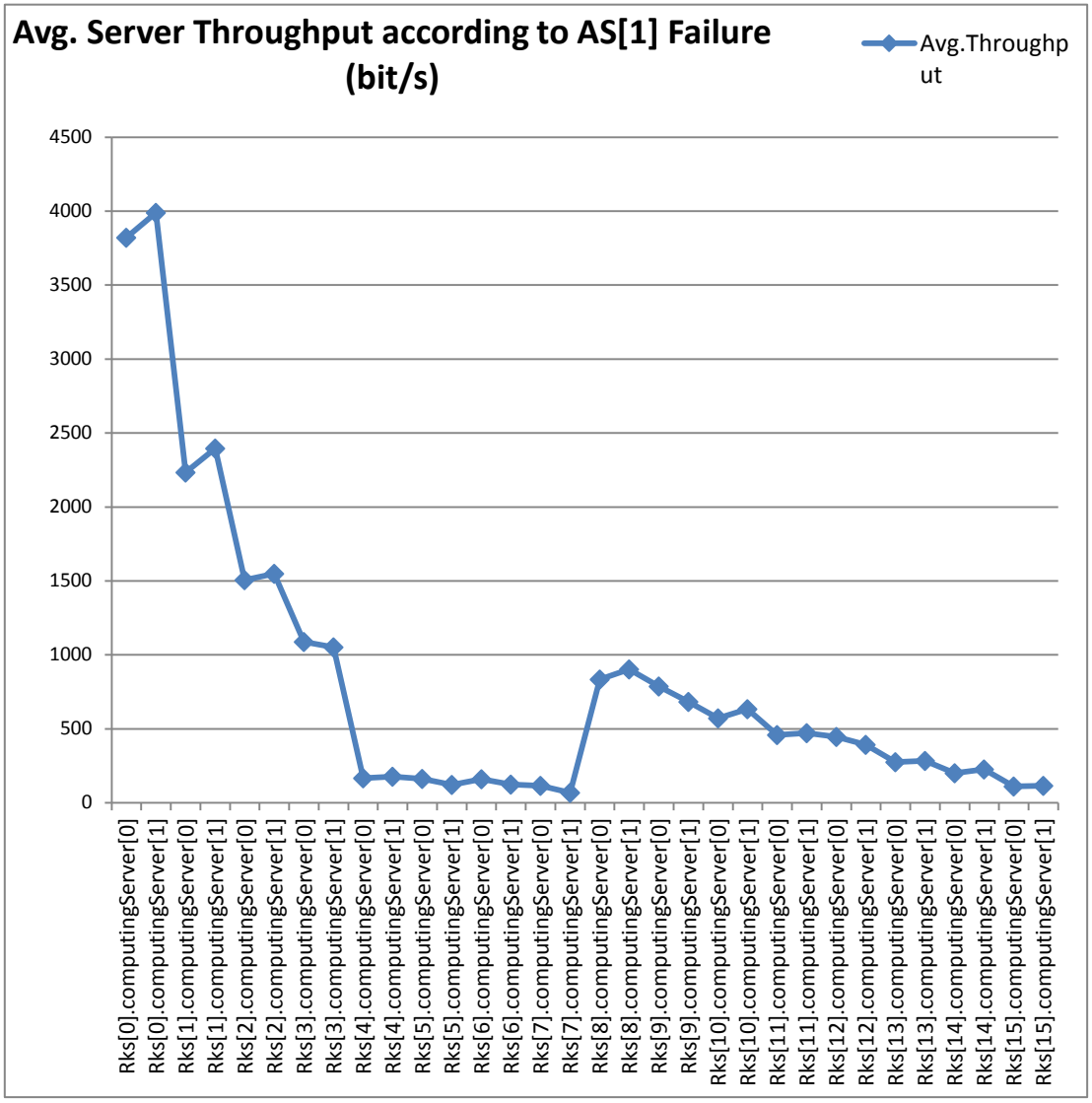


Figure 4.20 - Avg. server throughput according to AS[1] Failure (bit/s)

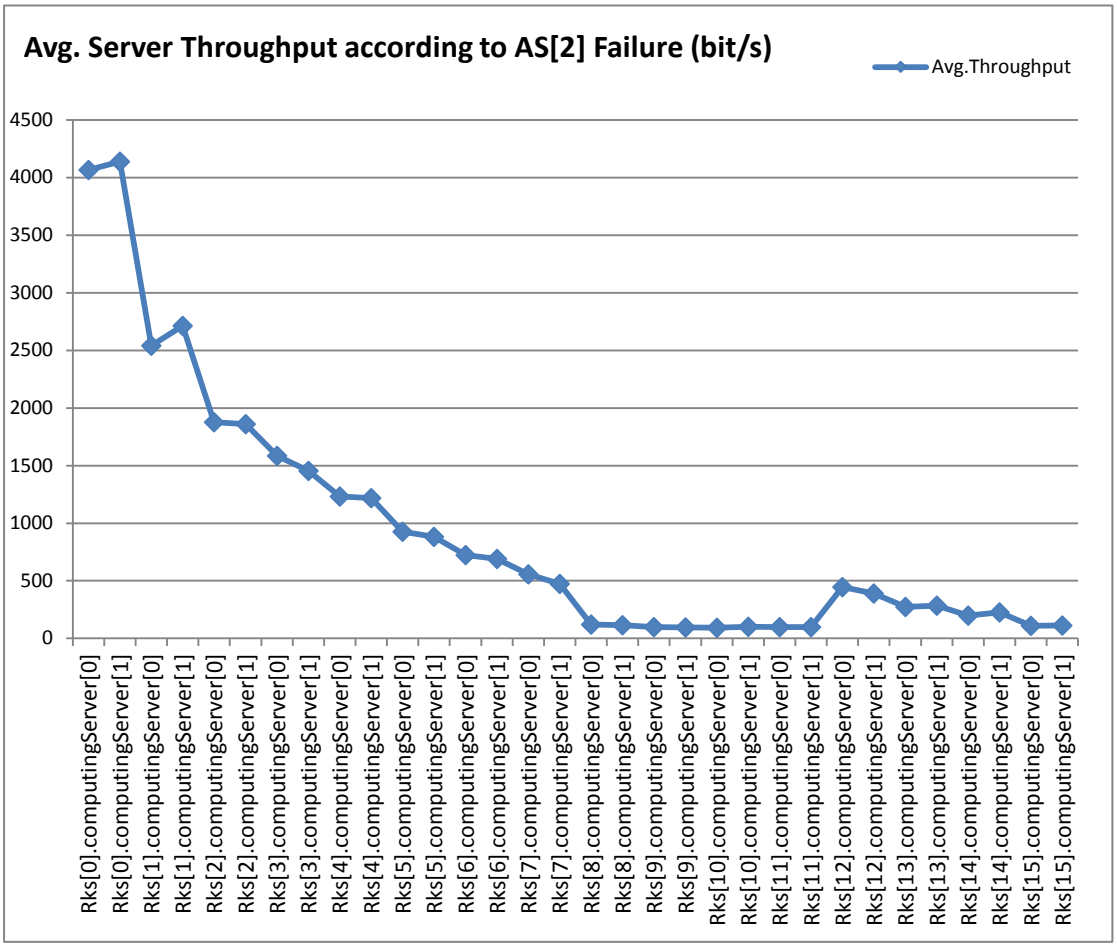


Figure 4.21 - Avg. server throughput according to AS[2] Failure (bit/s)

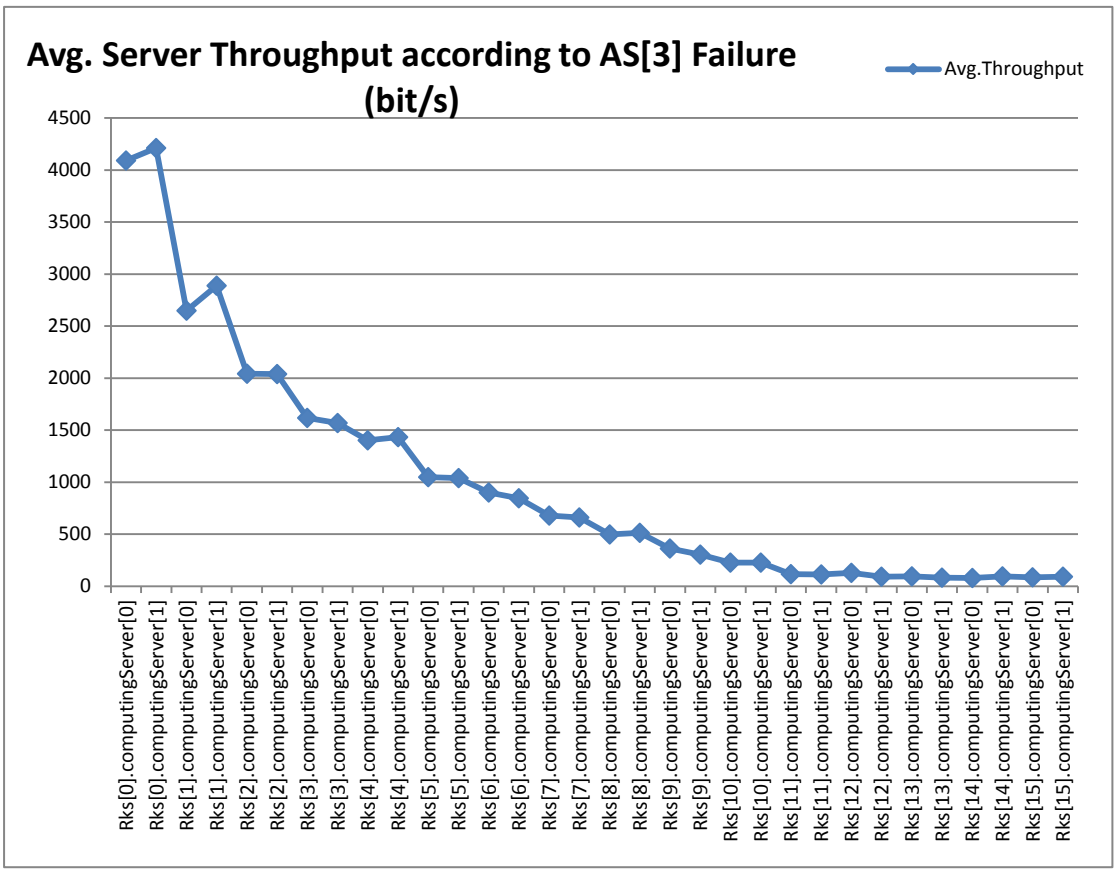


Figure 4.22 - Avg. server throughput according to AS[3] Failure (bit/s)

The switch failures reacted on server throughputs is depicted as the above figures, and failure on AS[0] lowered the throughput_rx of Rks[0-3], while AS[1] reacted on Rks[4-7]; AS[2] on Rks[8-11]; and AS[3] on Rks[12-15].

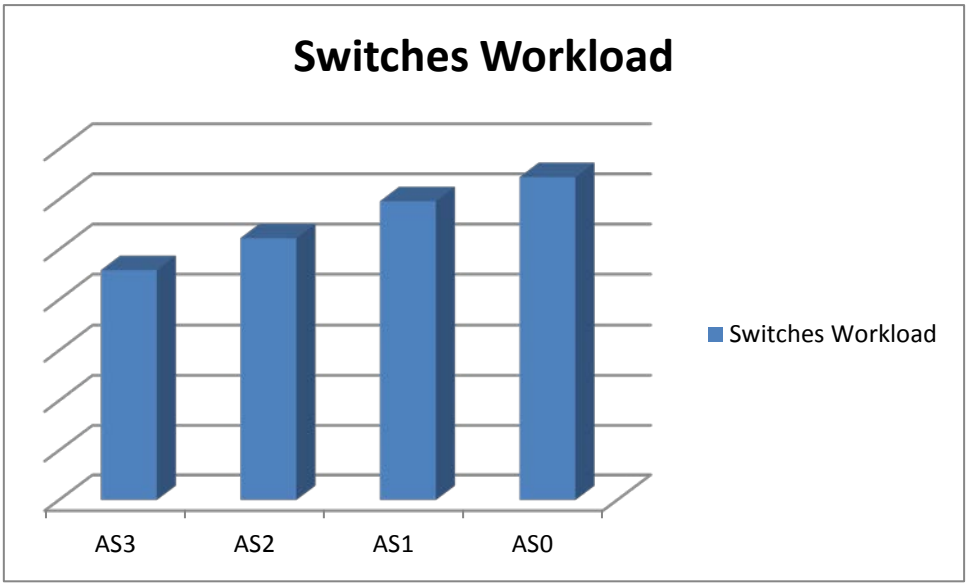


Figure 4.23 – Switch workload

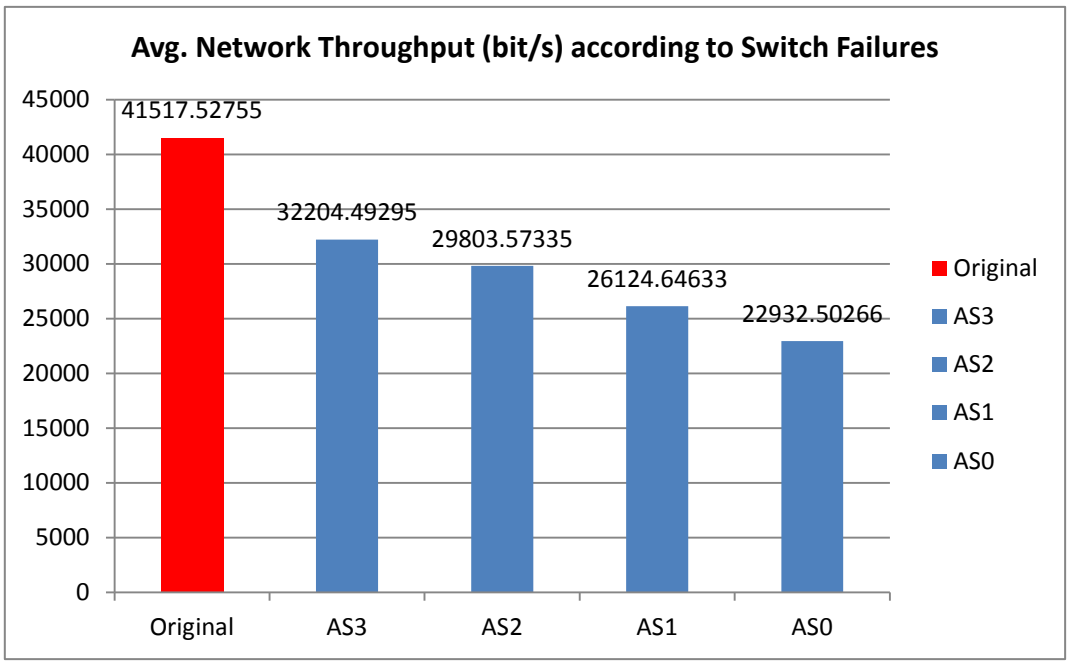


Figure 4.24 – Avg. network throughput (bit/s) according to switch failures

According to the corresponding relation with servers, AS[0] connected the highest loaded Racks which took the responsibility of transmitting the largest number of packets, while AS[3] received the least number of packets that were sent from the first 12 racks so that AS[3] had the lowest workload. The switch load was thus demonstrated as the above figure. Failure on AS[0] minimized the average network throughput of 44.76% from the initial network, where AS[3] degraded the average network throughput to 22.43%.

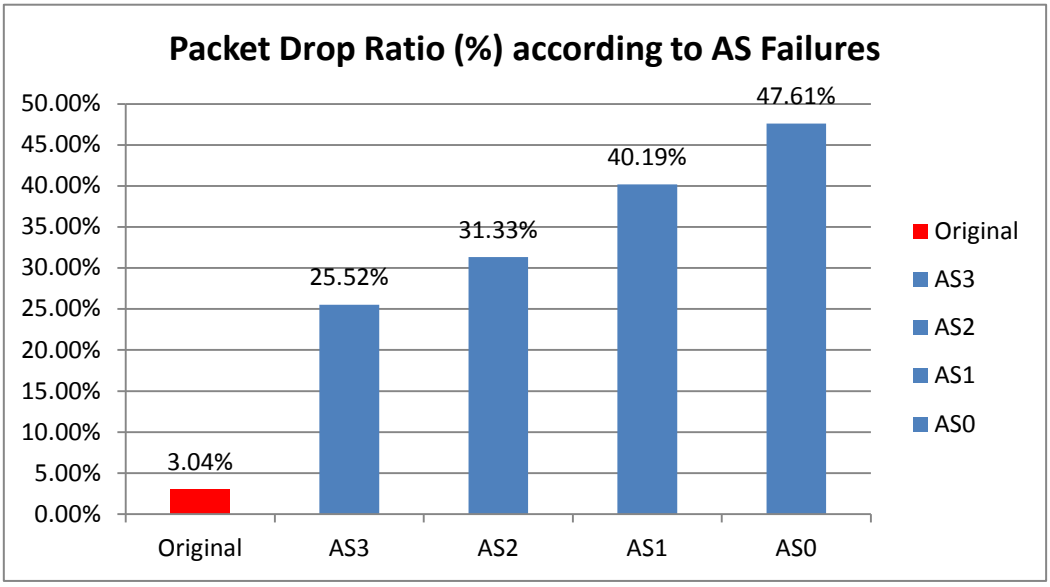


Figure 4.25 – Packet drop ratio (%) according to AS failures

The packet drop ratio based on switch failures demonstrated a reverse pattern from the one with average network throughput. This result predicted that all failures generated high PDR with an increasing pattern in the order of AS[3, 2, 1, 0]. In contrast to the server failures, switch failures produced much larger PDR, while despite that the Rks[0].server[0] failure dropped the largest number of total packets (15.57%). Among server failures, the failure of AS[0] dropped almost half the number of total packets.

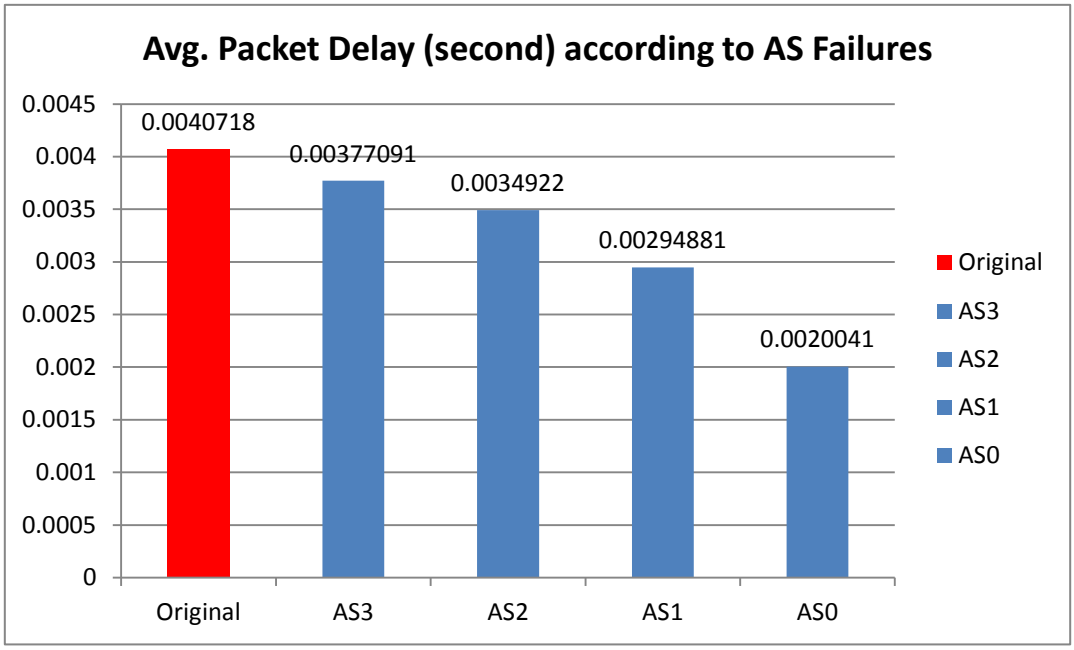


Figure 4.26 – Avg. packet delay (second) according to AS failures

Queuing time decreased when a large number of packets were dropped from the AS[0], and the average delay decreased to the lowest period as expected. With comparison to the server failures, more packets were dropped, causing the number of queuing packets to be reduced, so that decreased the average packet delay.

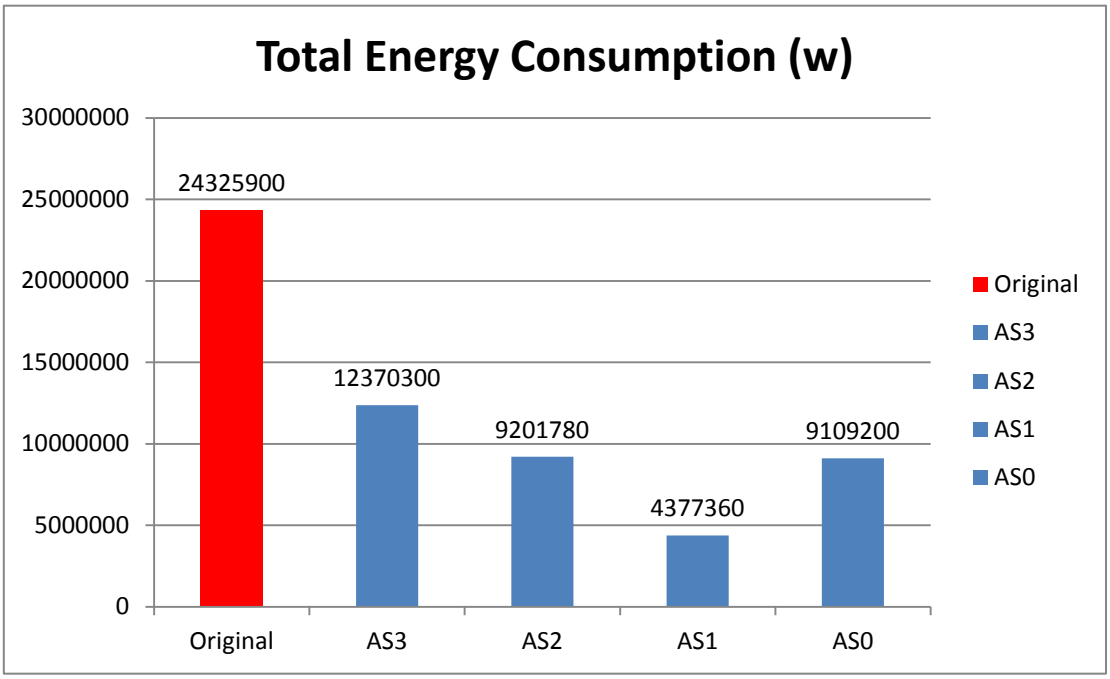


Figure 4.27 – Total energy consumption (w)

Servers transmitted packets to servers inside a rack directly through the corresponding edge router, but needed assistance of the edge router and connected aggregation switch to send and receive messages from neighboring racks, while core switch was necessary for transferring the packets to the farther servers. The failure on AS[0] disconnected two groups of edges, one with Edge[0, 1, 2, 3] and the other one between AS[0] and the core switches. The AS[0] failure caused the largest amount of data from far servers to be forced to be dropped at the core switch layer. In addition, the amount of packets dropped by connected edge routers from internal servers led to energy increments by a core switch and edge routers. This accounts for a reasonable portion due to the increased workload. This is the essential difference in energy consumption in contrast to the one with the other AS failure.

As aforementioned, the total energy consumption of a DC network can be expressed as:

$$E_T = P_{switch} + P_{server} \tag{4.4}$$

So it can be expanded as the following equation,

$$E_T = \sum_{i=0}^{n=2} P_{C_i} + \sum_{i=0}^{n=4} P_{A_i} + \sum_{i=0}^{n=16} P_{E_i} + P_S \tag{4.5}$$

Where P_{C_i} is the power consumption for the core switch i, P_{A_i} refers to the power consumption for the aggregation switch i, P_{E_i} represents the power supplied for edge router i, while P_S represents the total power consumption for servers.

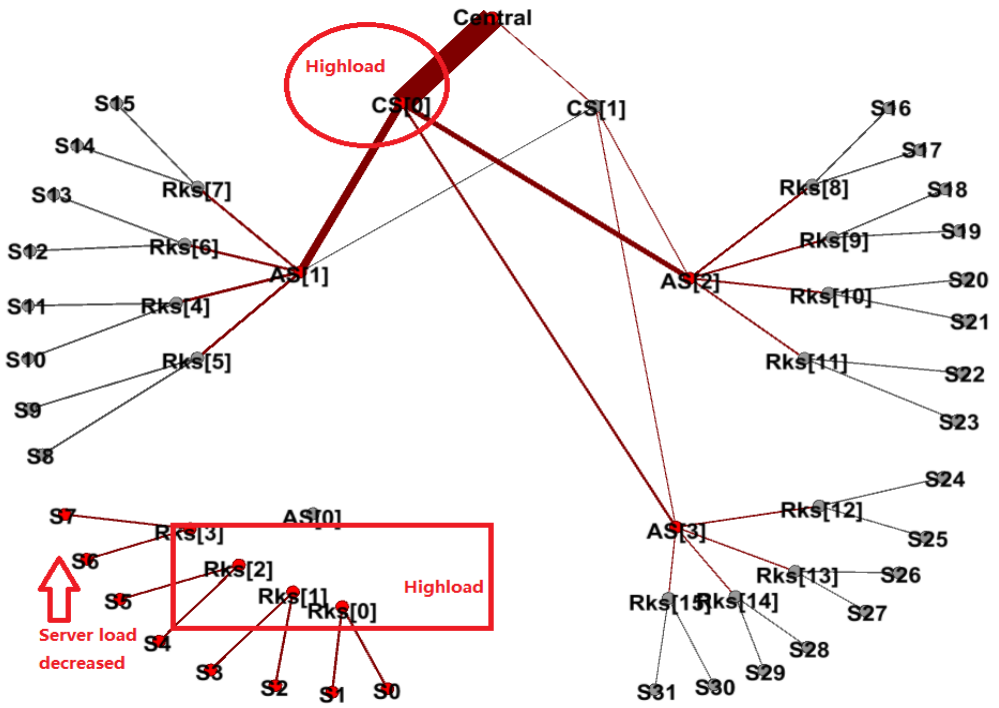


Figure 4.28 – Component traffic load vs. energy change illustration

The total energy consumption of the network after the failure of AS[0] is:

$$E_{T\prime} = \{(P'_{C_0} + \delta) + P_{C_1}\} + \left(P_{A_{0_{fix}}} + \sum_{i=1}^{n=3} P_{A\prime_i}\right) + \left(\sum_{i=0}^{n=4} P_{E\prime_i} + \sum_{i=4}^{n=12} P_{E_i}\right) + (P_S - \varepsilon) \quad (4.6)$$

The failure of AS[0] disconnected Racks[0-3], thus the packets could not be received by Server[0-7]; in the same way, packets also could not be sent away from Server[0-7] so that the other servers had less tasks received to be computed. This significantly decreased the total energy consumption especially servers[0-7] which took the responsibility for the heaviest computation tasks. However, even though Rack[0], Rack[1], Rack[2], and Rack[3] lost the capacity of transmitting packets to each other, the two servers with corresponding edge routers in each Rack[0-3] constituted a small LAN which could only interact with each other because the AS[0] failure did not affect the edge router in each rack, as shown in Table 4.13, Edge[0-3]

had received 800 packets, i.e., server[0] could only sent 400 packets to server[1] and vice versa, but server[0] could not send and receive packets outside Rack[0].

	rcvdPk-Original	**rcvdPk-after AS[0] failure**	**Difference**
AS[0].pppg[0]	0	0	6332
AS[0].pppg[16]	2194	0	
AS[0].pppg[17]	1692	0	
AS[0].pppg[18]	1346	0	
AS[0].pppg[19]	1100	0	
AS[1].pppg[0]	2183	0	2183
AS[1].pppg[16]	1028	1028	
AS[1].pppg[17]	825	825	
AS[1].pppg[18]	689	689	
AS[1].pppg[19]	620	620	
AS[2].pppg[0]	2595	1342	1253
AS[2].pppg[16]	480	480	
AS[2].pppg[17]	408	408	
AS[2].pppg[18]	325	325	
AS[2].pppg[19]	237	237	
AS[3].pppg[0]	2716	1819	897
AS[3].pppg[16]	210	210	
AS[3].pppg[17]	135	135	
AS[3].pppg[18]	94	94	
AS[3].pppg[19]	30	30	
Edge[0]	2590	800	5928
Edge[1]	2492	800	
Edge[2]	2146	800	
Edge[3]	1900	800	
Edge[4]	1828	1828	
Edge[5]	1625	1625	
Edge[6]	1489	1489	
Edge[7]	1420	1420	
Edge[8]	1280	1280	
Edge[9]	1208	1208	
Edge[10]	1125	1125	
Edge[11]	1037	1037	
Edge[12]	1010	1010	
Edge[13]	935	935	
Edge[14]	894	894	
Edge[15]	830	830	

Table 4.13 - The change of number of packet received after AS[0] failure for specific node port

Due to the failure of AS[0], CS[0] bore a heavier workload of disposing dropped packets; the increment of energy δ was added to the power consumed by CS[0], so that the total energy consumption for the core layer switches was recalculated as $\{(P_{C\prime_0} + \delta) + P_{C_1}\}$ where P_{C_0} represents the original power consumption for CS[0], $P_{C\prime_0}$ stands for the new power consumption based on less packets received, and P_{C_1} stands for the power consumption of CS[1] which is fixed. In addition, the failure caused no responses to AS[0], but there is a great probability that AS[0] still consumed a fixed amount of energy $P_{A_{0_{fix}}}$ due to the fixed consuming components but less than while in a working state. The rest of the aggregation switches also consumed less power because the failure of AS[0] resulted in the number of packets received by other aggregation switches to decrease but only by a small portion. However, the connected four edge routers bore a much heavier workload of disposing the packets which got stuck as a result of the situation of CS[0]. The power consumption based on received packets for Edge[0, 1, 2, 3] is represented as $\sum_{i=0}^{n=4} P_{E\prime_i}$ and the energy of bearing load on dropping packets is γ. Moreover, the computing power for the packets not received at Server[0-7] would be avoided, which refreshes the server power consumption to $(P_S - \varepsilon)$.

Therefore, based on the failure of AS[0], the difference ΔE_{A_0} between $E_{T\prime_{A_0}}$ and $E_{T_{A_0}}$ would be

$$\Delta E_{A_0} = \delta + \left(P'_{C_0} - P_{C_0}\right) + \left(P_{A_{0_{fix}}} - P_{A_0}\right) + \left(\sum_{i=1}^{n=3} P_{A\prime_i} - \sum_{i=1}^{n=3} P_{A_i}\right) + \gamma + \left(\sum_{i=0}^{n=4} P_{E\prime_i} - \sum_{i=0}^{n=4} P_{E_i}\right) - \varepsilon \quad (4.7)$$

Due to the received packets reduced, $\sum_{i=1}^{n=3} P_{A\prime_i} - \sum_{i=1}^{n=3} P_{A_i} < 0$, $\sum_{i=0}^{n=4} P_{E\prime_i} - \sum_{i=0}^{n=4} P_{E_i} < 0$, $P'_{C_0} - P_{C_0} < 0$, energy on CS[0]`s extra dropping packet load bearing $\delta > 0$, AS[0] power consumption $P_{A_{0_{fix}}} - P_{A_0} < 0$, energy on Edge[0-3]s` extra dropping packet load bearing $\gamma > 0$, and $-\varepsilon$ represents the server energy reduction accounts for the largest portion in ΔE_{A_0} must be negative value. As a result, ΔE_{A_0} is

negative but the value is varying based on the difference between the value of positive sum and the value of ε.

In the same way, every variable in ΔE_{A_3} ΔE_{A_2} ΔE_{A_1} are different from the one in ΔE_{A_0} because the traffic pattern varied widely. Therefore, each AS failure generated as different energy consumption result. However, the failures of AS[3, 2, 1] produced a ladder-like downward decrease on total power consumption where AS[3] decreased by 49.15% compared to the original network: AS[1] at 82.01% and AS[0] at 62.55%, respectively.

Switch Failure	**AS[3]**	**AS[2]**	**AS[1]**	**AS[0]**
Energy degradation ratio from original	49.15%	62.17%	82.01%	62.55%

Table 4.14 - Energy degradation ratio for various AS failure

4.2.14 Case 1 - Scenario 3: Network Performance according to Server/Switch/Rack Failures

Notation in result figure	**Representation**
Original	Network with no failures
SF15	Server Failure of Rack[15].server[1]
SF0	Server Failure of Rack[0].server[0]
RF	Rack Failure (Rack[0])
AS	Aggregation Switch Failure (AS[0])

Table 4.15 - Notice on notation representation for the use in later results

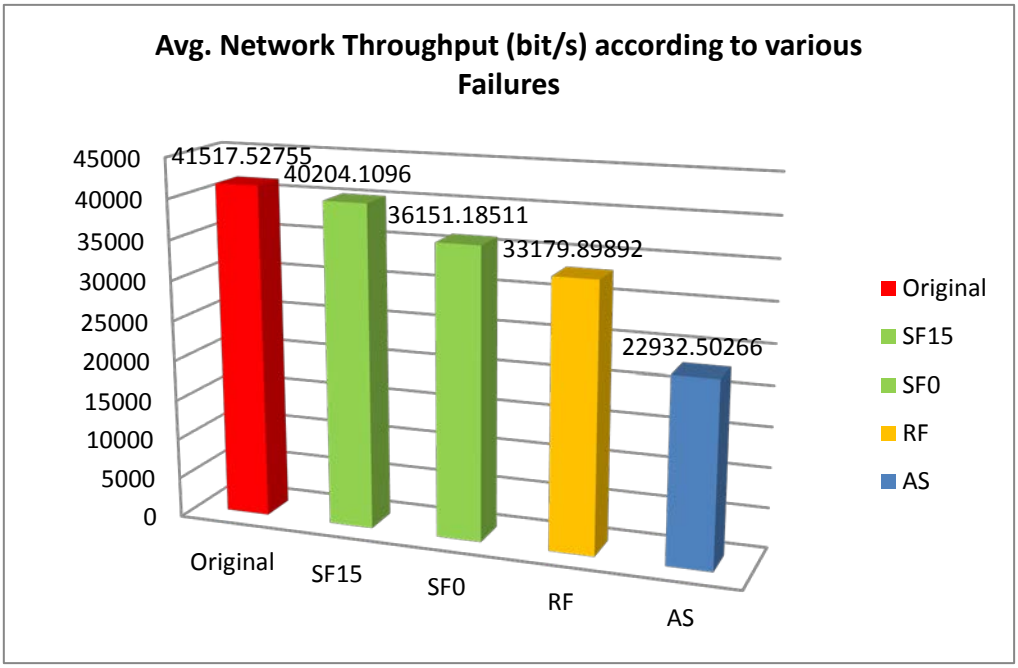


Figure 4.29 - Avg. network throughput (bit/s) according to various Failures

Server Failures seem to produce the smallest effect on the network throughput where the failure of a low-load server (Rack[15].server[1]) only decreased 3.16% of the entire average network throughput, failure of high-load server (Rack[0].server[0]) decreases the avg. network throughput to 12.92%. The rack failure was selected from Rack[0], the one server[0] belonged to, which is also the rack that bore the highest workload, which resulted in a decrease of 20.08% to the average network throughput. Furthermore, as AS[0] was the most important node in this network, its failure had the largest impact on the average network throughput compared to server failure and rack failure.

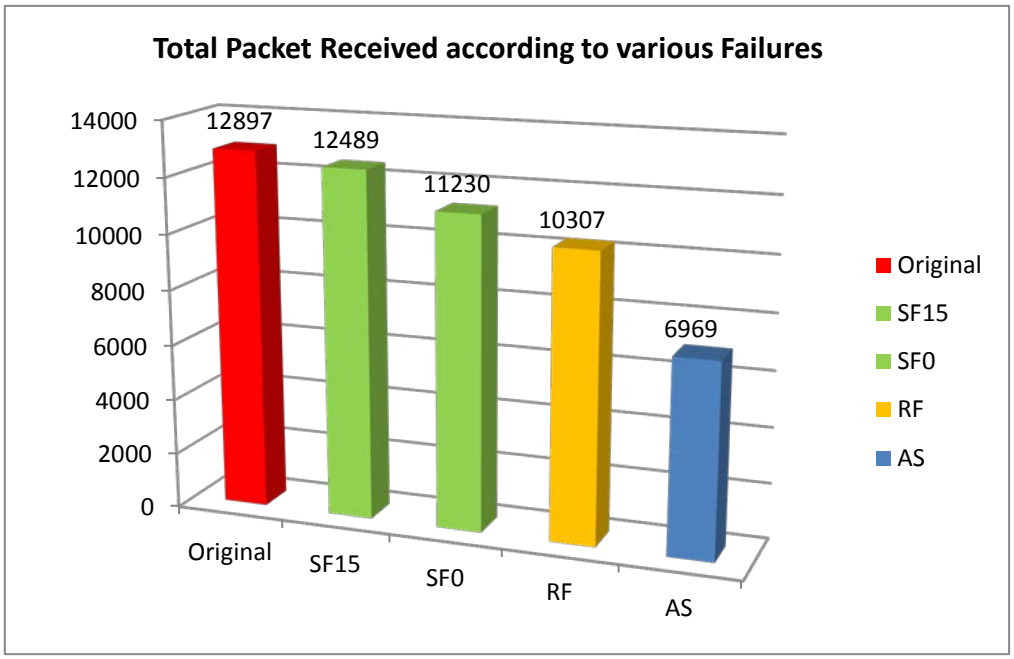


Figure 4.30 - Total packet received according to various Failures

Similar to the result of average network throughput, the number of packets received decreased in the order of SF15, SF0, RF and AS. The failure of AS[0] reduced almost half the number of packet received, and this significantly influenced the whole network performance no matter how the QoS was defined in the designated SLA.

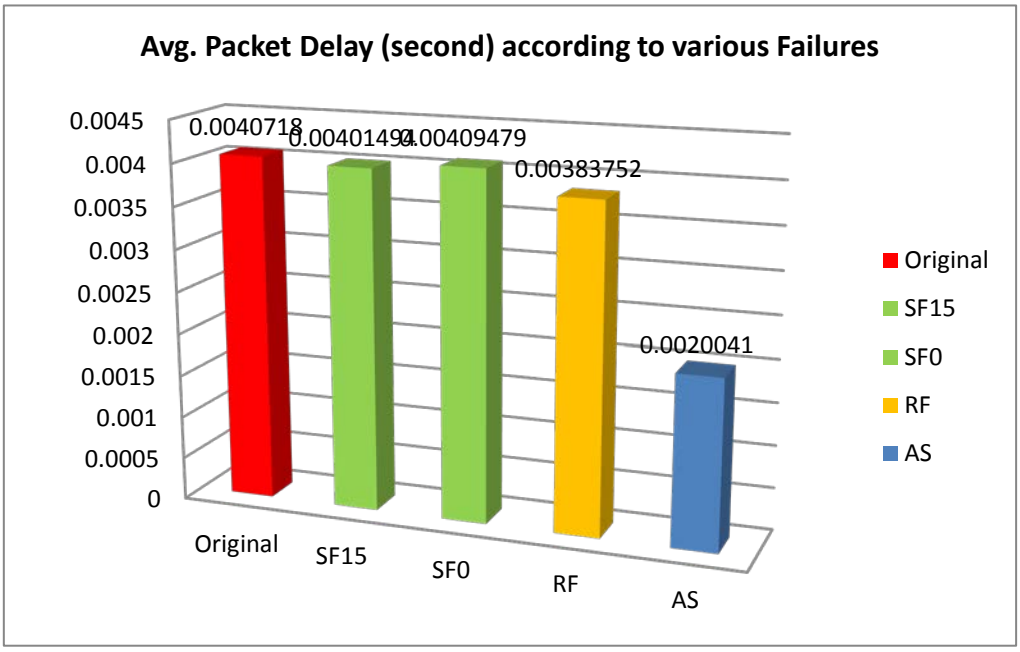


Figure 4.31 - Avg. packet delay (second) according to various Failures

The server failure and rack failure did not influence the average packet delay critically, which was still around 0.004s. Although the failure of AS[0] reduced the

average time of packet transmitting from source node to destination node to 0.002s, the number of packet received also halved.

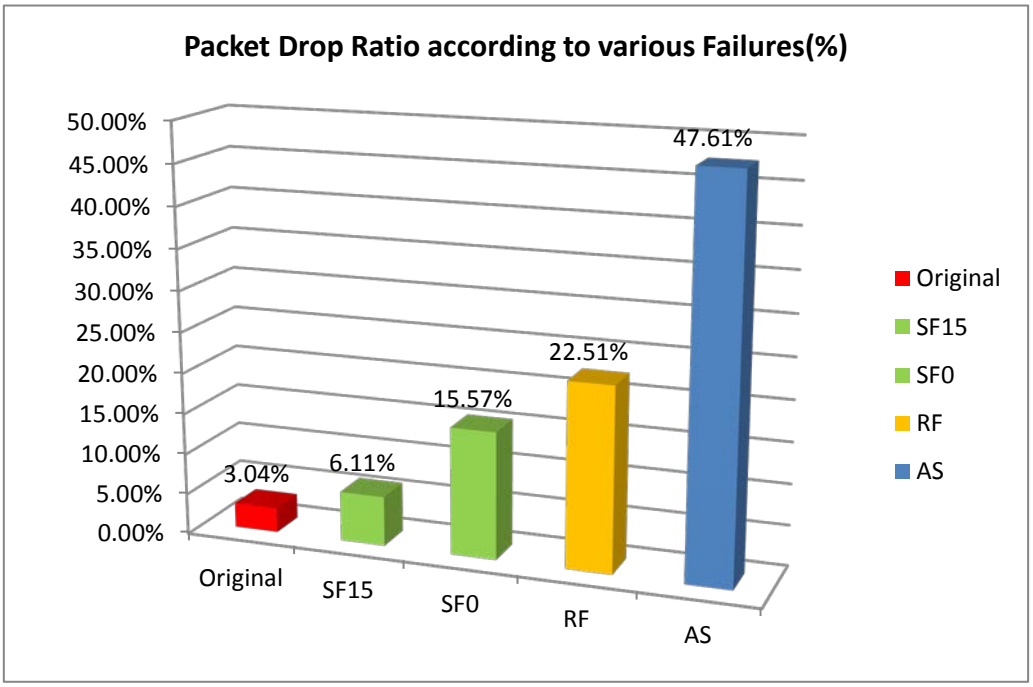


Figure 4.32 - Packet Drop Ratio (%) according to various Failures

The Rack[0] failure enlarges the network packet drop ratio (22.51%) compared to server failure, particularly to Rack[15].server[1], i.e., 6.11%. Nevertheless, the AS[0] failure doubled the packet drop ratio produced by the rack failure, i.e., 47.61%, and lowered the network performance by almost half.

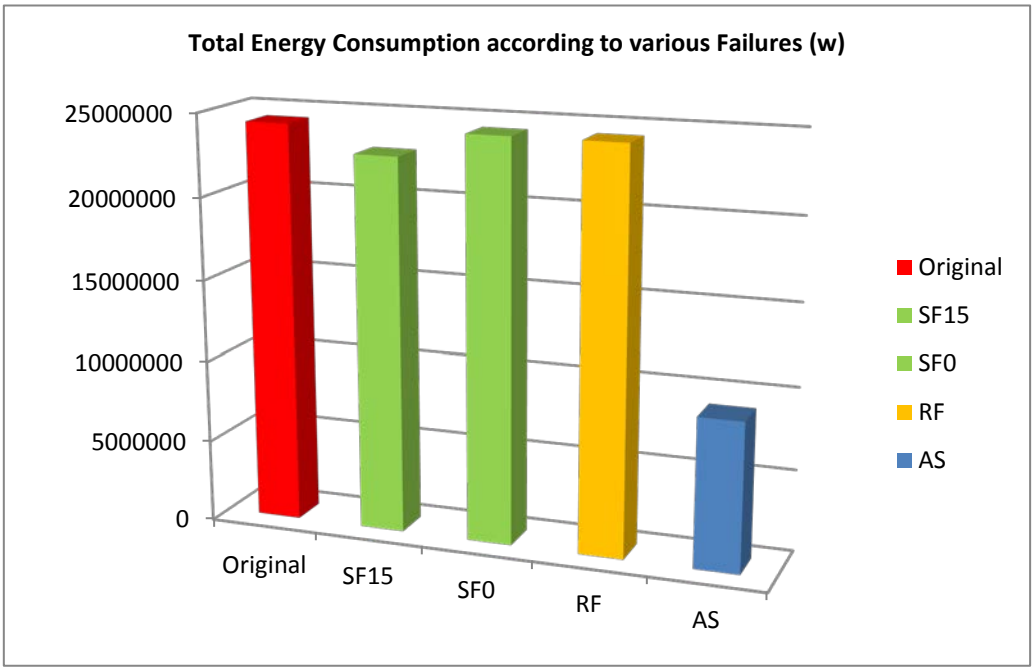


Figure 4.33 - Total energy consumption according to various Failures (w)

As aforementioned, the total energy consumption change would took several factors into consideration which included the switch load bearing energy, the server energy reduction, and the switch energy change according to the traffic change. The server energy change ε plays the crucial role in the total energy change, where the AS[0] failure disconnected eight servers so as to decrease the total energy consumption significantly. On the other hand, the energy consumed for the network did not change much when the rack is failed, because the absolute value of the increase of power supplied for switch load remained almost the same as the absolute decrease value for the energy for handling jobs in switches, plus the server energy reduction.

$$\Delta E_{A_0} = \delta + \left(P'_{C_0} - P_{C_0}\right) + \left(P_{A_{0_{fix}}} - P_{A_0}\right) + \left(\sum_{i=1}^{n=3} P_{A'_i} - \sum_{i=1}^{n=3} P_{A_i}\right) + \gamma + \left(\sum_{i=0}^{n=4} P_{E'_i} - \sum_{i=0}^{n=4} P_{E_i}\right) - \varepsilon \quad (4.8)$$

4.3 Case 2: DCNs` Performance vs. LFR

4.3.1 Simulation setup

In order to make the simulation more realistic, in this case, we deployed a different traffic pattern compared to the first study. The unbalanced traffic pattern is regarded as the closest situation to the DCN traffic as can be realistically simulated. This movement made the simulation more complicated to conduct, but valuable for its research contribution due to there not being much literature that reveals the DCN performance even under normal daily DCN traffic conditions.

In this study, we built an energy-aware distributed DCN through simulation studies in order to observe the impact of the underlying network connectivity on the DCN`s performance. We modeled four identical datacenters distributed in geographical fashion by using the CloudNetSim++ simulator, and the datacenters were interconnected by Full-mesh topology to maximize the network stability. Five types of DCN architecture were simulated which were Fat-tree, BCube-2 layers (shown as BCube1 in results), BCube-3 layers (shown as BCube2 in results),

Three-tier, and HyperFlatnet. UDP transportation protocol was used and the traffic was still followed all-to-all but with an unbalanced pattern. Servers still generated the same number of packets but the receivers were organized randomly.

Inter-DCN	Full-mesh Network
Intra-DCN	Fat-tree; BCube-2 layers (shown as BCube1 in results); BCube-3 layers (shown as BCube2 in results); Three-tier; HyperFlatnet
Traffic Protocol	User Datagram Protocol (UDP)
Traffic Type	(all-to-all) All servers to all servers
Traffic Pattern	Unbalanced, tends to real-life cases
Packet Size	2500 bytes
Packet Send Interval	0.8s
Connection Type	Gigabit Ethernet links
Link quality	Packet Error Rate = 0.8% (0.8 packet will be wrong when transmits 100 packets through one link)
Link Bandwidth	Varies from 1Gbps to 10Gbps inside intra-DCN; 100Gbps for inter-DCN
Simulation Time	2000 seconds

Table 4.16 - Case 2 - Network setup

4.3.2 Topology setup

In order to make network performance comparable between each data center topology, the number of servers and the inter-DCN were maintained as fixed, and as a consequence, the variance then appeared in intra-DCN, which arose from the DCN

type and the internal architecture such as DCN layers, the number of switches and links, the server port number, and the switch port number.

	Fat Tree	**Three Tier**	**BCube1**	**BCube2**	**HyperFlatNet**
Switch degrees	3 layer	3 layer	2 layer	3 layer	2 layer
Switches Breakdown	Core=16 Aggregation=32 Edge=32	Core=16 Aggregation=32 Edge=32	Level0=32 Level1=32	Level0=48 Level1=48 Level2=48	Internal=64 External=64
No. of Switches	80	80	64	192	128
No. of Servers	256	256	256	256	256
No. of Links (4 distributed DC)	384	448	512	512	976
No. of Links (mesh)	16	16	16	16	16
Total Links	400	464	528	528	992
Core layer Switch ports	4	8	-	-	-
Aggregation layer Switch ports	4	6	-	-	-
Edge layer Switch ports	10	10	-	-	-
Level 0 Switch ports	-	-	8	4	-
Level 1 Switch ports	-	-	8	4	-
Level 2 Switch ports	-	-	-	4	-
Internal Switch ports	-	-	-	-	5

…continued					
External Switch ports	-	-	-	-	4
Max. Server ports	1	1	2	3	2

Table 4.17 - Case 2 – DCNs Topology setup

4.3.3 Topological comparison

When the number of servers was maintained as 256 as highlighted in the above table, there were 3 layers of switches in Fat Tree, Three Tier and BCube-2 level, as a result, these three DCNs possessed more number of switches than the other 2 layers DCNs (BCube-1 level & HyperFlatNet).

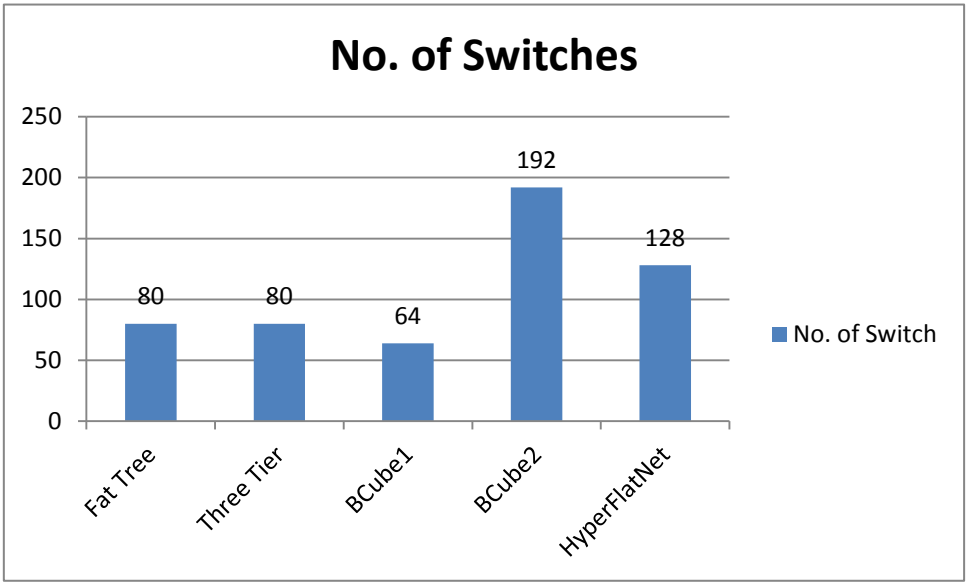


Figure 4.34 - Number of switches comparison between topologies

However, due to the connecting patterns varied diversely, difference appeared on the number of links used, which the HyperFlatNet consumed the maximum number of links (976) while Fat Tree obtained the least number of links which was 384.

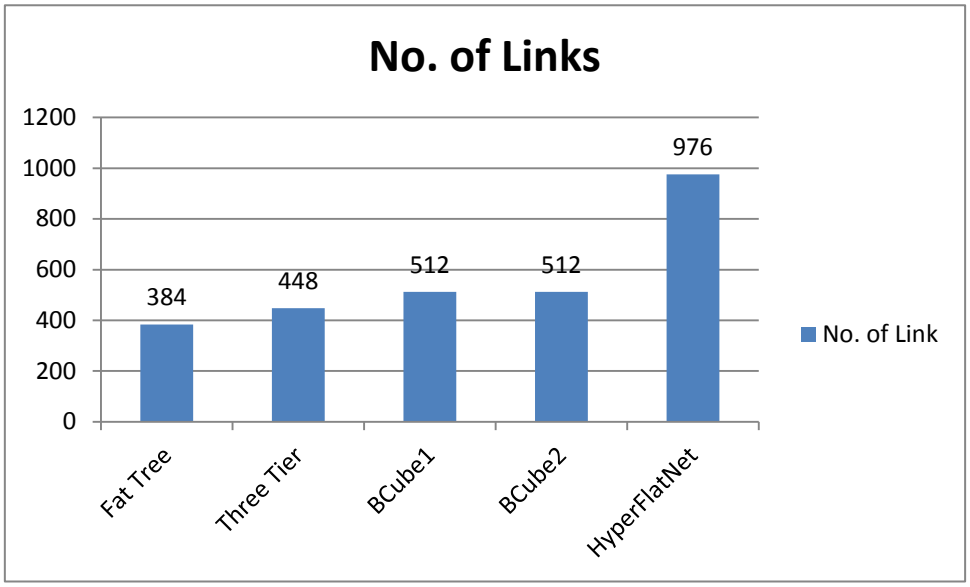


Figure 4.35 - Number of links comparison between topologies

On the other side, although 2 BCubes were different from the switch layers and the switch numbers, the links used were maintained as the same because both the switch type and server type differentiated between port numbers: BCube-1 level employed 8-port switches and only 2-port servers, while BCube-2 level applied comparatively less costly 4-port switches and 3-port servers.

Fat-tree and Three-tier DCN have very similar architectures which only vary between the numbers of links consumed and switch types, as discussed in the previous section. Fat-tree has less number of link oversubscription issues than the mainstream Three-tier DCN, which caused the fewer links used compared to Three-tier. Also, from the view of DCN arrangement, although Fat-tree and Three-tier obtain the same number of racks (32 racks in this scenario), the cost of establishing Fat Tree DCN can be saved from not only the number of links consumed, but the number of maximum-port switches used, as the core and aggregation layer switches can be substituted to commodity 4-port switches.

4.3.4 Case 2 - Scenario 1: Network Performance of Various DCNs with 0% LFR

In this section, we compare the five DCNs under a 0% link failure ratio. The results show the original network performance and the evaluation generated a comprehensive analysis of these topologies.

4.3.5 Results Analysis for Scenario 1

The following table shows the network performance and energy consumption for various DCNs.

DCN	Energy	Network latency (ms)	Avg. Network Throughput (bits/s)	Total Packets Received	Packet Drop Ratio
Fat-tree	23289.3	8.5051	154383.375	2007	3.238%
HyperFlatNet	22842.4	5.09346	104444.568	1965	1.157%
BCube1	39038.3	0.54233	100442.854	1929	0.942%
BCube2	55378.4	1.06712	98117.726	1879	0.9404%
Three-tier	22617.7	10.0481	154215.035	2004	3.311%

Table 4.18 - Network Performance & Energy Consumption comparison for different DCNs with no failures

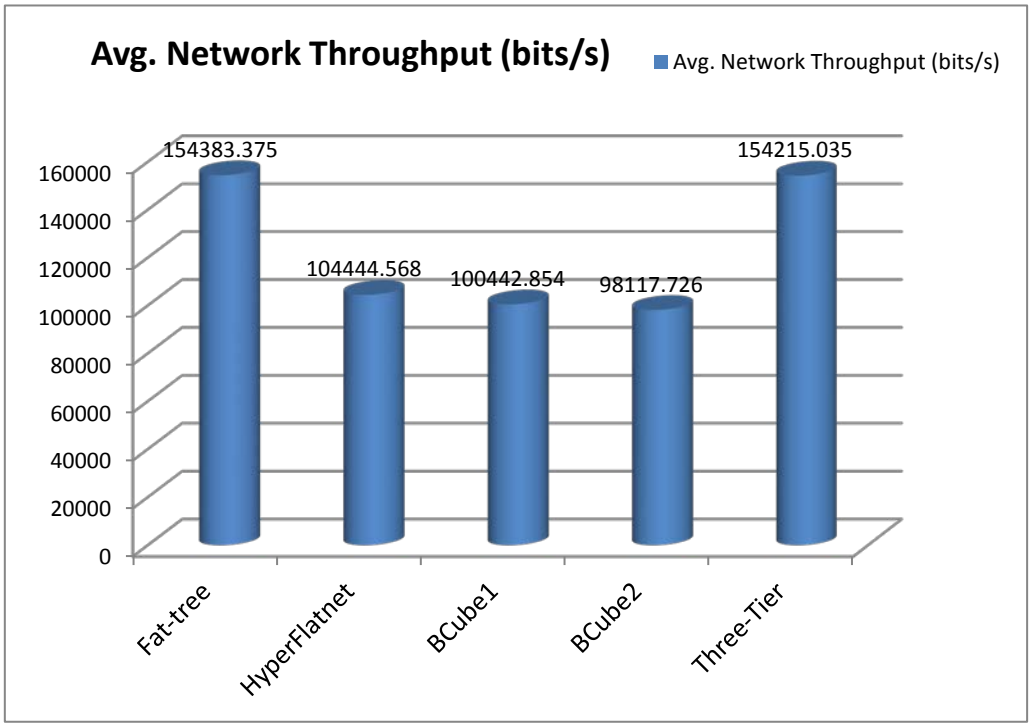


Figure 4.36 - Avg. network throughput (bits/s) comparison for different DCNs with no failures

The tree-based DCN topologies (Fat-tree and Three-tier) achieved the highest average network throughput which was approximately 1/3 higher than the server-centric topologies (HyperFlatNet, BCube-2 layers and 3 layers). The topological aspect significantly determined the network performance because they followed the same traffic pattern. The tree-based topologies used the least network components such as switches and links, but gained the highest network throughput.

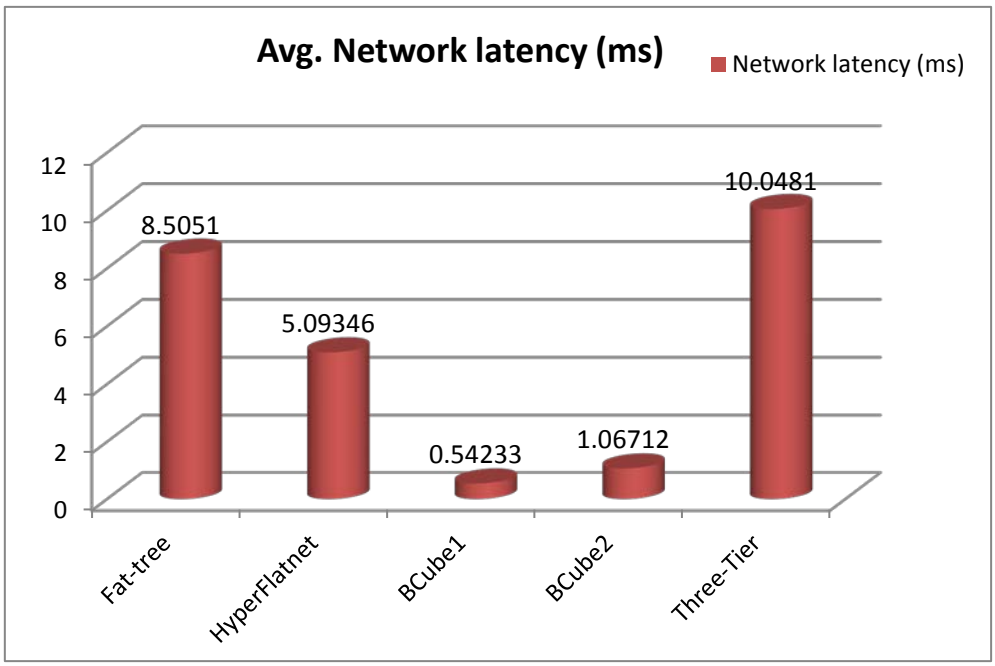


Figure 4.37 - Avg. Network latency (ms) comparison for different DCNs with no failures

BCube-2 layers and 3 layers achieved the lowest network latency, and HyperFlatNet also gained very reasonable latency, which indicates the server-centric DCNs are definitely relaying messages quickly in comparison to the "switch-based" (tree-based) DCNs. The conventional Three-tier architecture produced the highest network latency due to the architectural property (switch-based) and the more network components (links) used compared to Fat-tree.

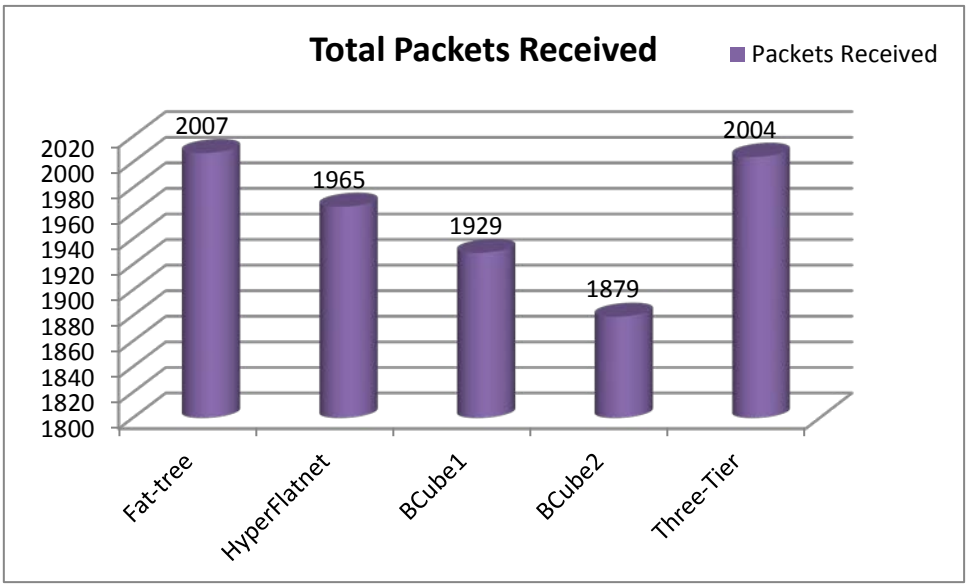


Figure 4.38 - Total number of packet received comparison for different DCNs with no failures

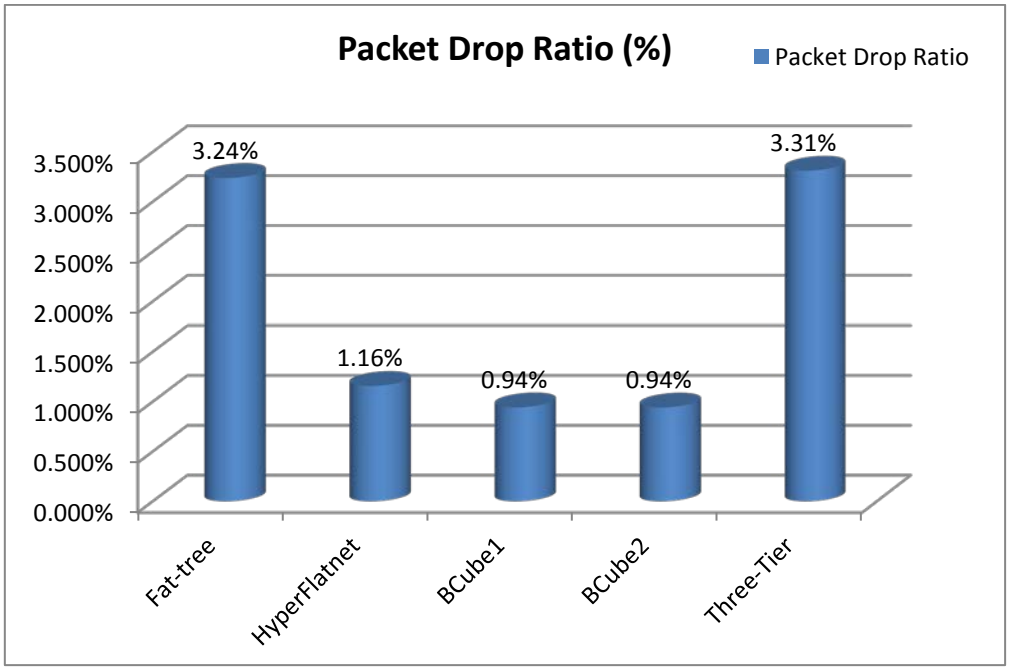


Figure 4.39 - Packet drop ratio (%) comparison for different DCNs with no failures

The result of total packets received and the packet drop ratio presented a similar pattern to the one for average network throughput. BCube-3 layers dropped most packets and Fat-tree and Three-tier performed the robustness characteristics in this movement.

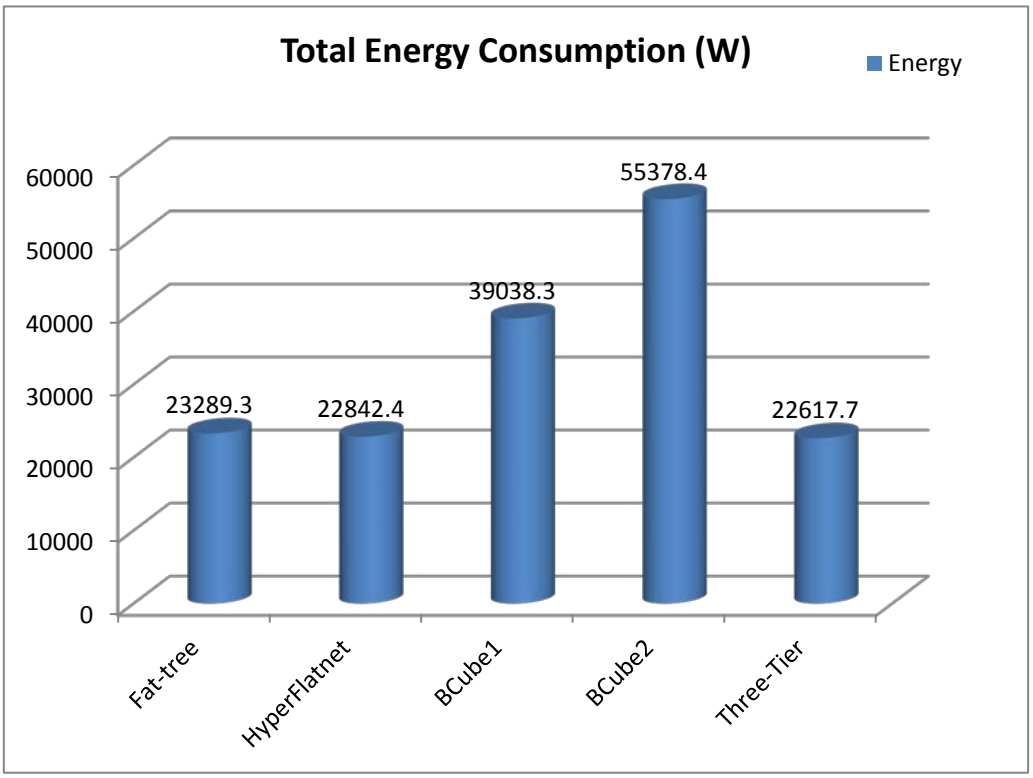


Figure 4.40 - Total energy consumption (w) comparison for different DCNs with no failures

BCube-3 layers consumed the largest amount of power, except for server power consumption. The power for supporting the largest number of switch (192) accounted for a relative high proportion of the total power consumed.

4.3.6 Case 2 – Scenario 2 (preparation): Link selection determination according to the Centrality metrics

In order to move on to the next process, LFR metric had to be determined, as the link failure selection would be a critical mission in this case. With the purpose of closely evaluating the DCN performance, prominent failures must be considered a top priority so that the effect will be enlarged according to the increasing LFR for the process of observing phenomenon. To accomplish this goal, we followed the methodology based on the first study, using centrality metrics to determine the most important nodes in the network. After that, a second round determination was conducted for deciding the important links based on the traffic load determined, for use in the later results analysis process.

For example, in the preparation process, we imported the datasheet of the five DC architectures into Gephi, calculated each architecture`s centrality value, and the decision was made after the result analysis. The centrality results of Fat-tree are shown below,

Notation	Representation
*	Implies all
R	Represents R*, Routers in full mesh network
CS[0]	The first core switch
AS[0,2,4,6]	The first, third, fifth and seventh aggregation switch
AS[1,3,5,7]	The second, fourth, sixth and eighth aggregation switch

Table 4.19 - Notice on notation representation for later use in the result

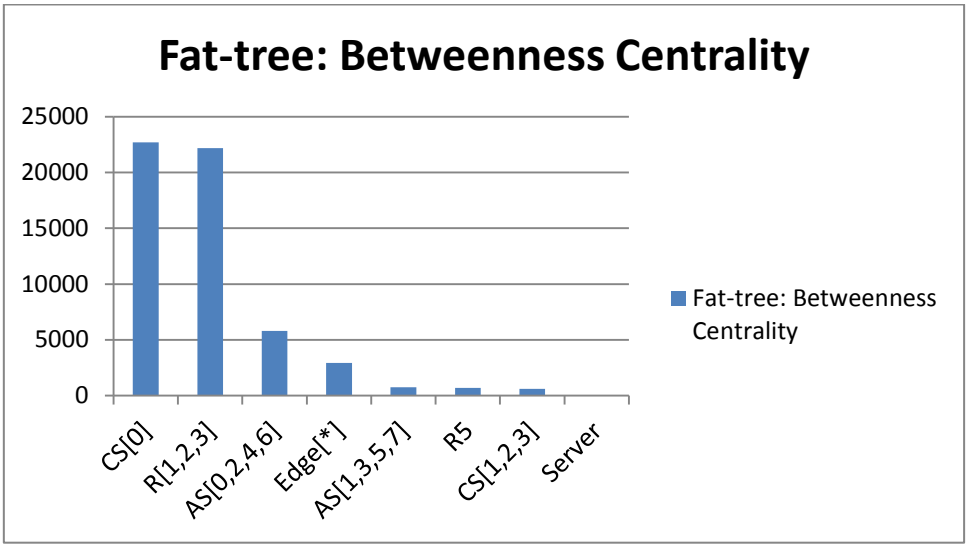


Figure 4.41 - Node betweenness centrality result for Fat-tree

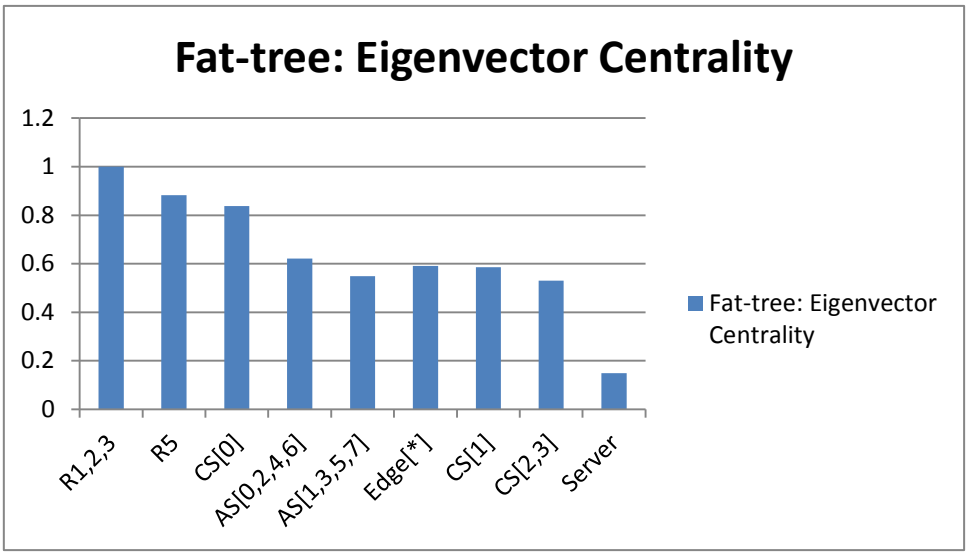


Figure 4.42 - Node eigenvector centrality result for Fat-tree

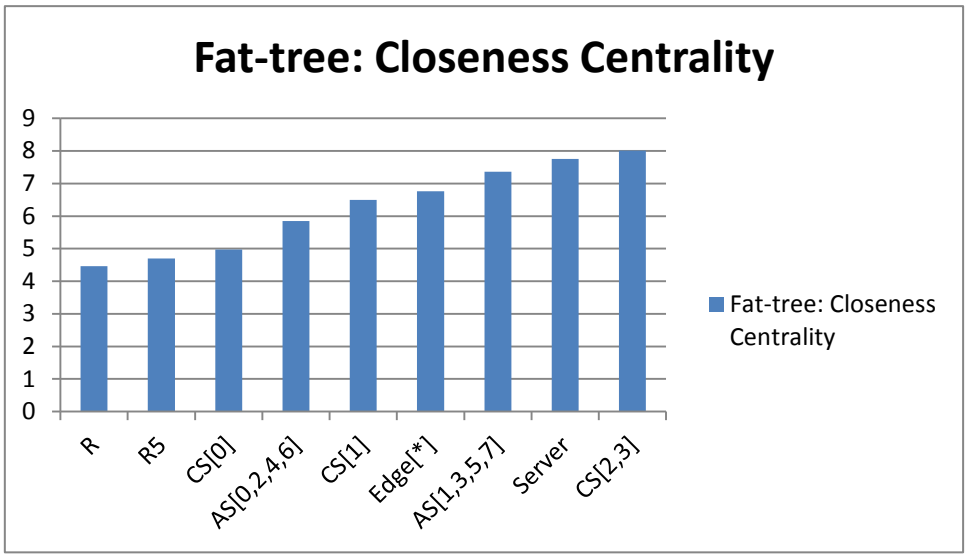


Figure 4.43 - Node closeness centrality result for Fat-tree

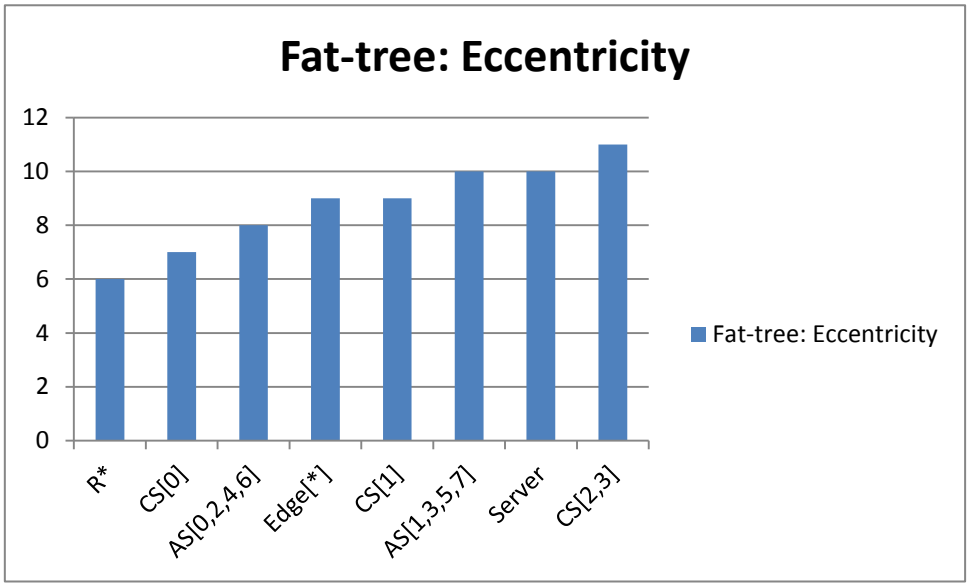


Figure 4.44 - Node eccentricity distribution result for Fat-tree

According to the node centrality result of the Fat-tree, CS[0] and AS[0,2,4,6] are considered the most important nodes. They gained the highest betweenness centrality and eigenvector centrality values, and the lowest closeness centrality and eccentricity value. Routers in mesh network should not be considered as critical nodes for Fat-tree. By following this inference, the links were randomly selected as the edges which belong to the critical nodes. In the case of Fat-tree, a set of edges connected to AS[0] were selected as the link failures. For BCube-2 layers, the higher layer switches (L1) were selected. For BCube-3 layers, L2 switches were selected. For HyperFlatNet, internal switches (ins) were selected. For Three-tier, AS and CS were selected.

The following tables are reference for the later process the researcher used to represent the network robustness metrics. They changed according to the increasing link failures. The figures in red indicate the critical links based on the links selection after consideration of the traffic load, which is shown as the following figures, representing the packet received from higher layer by the source node ports with LFR = 0%.

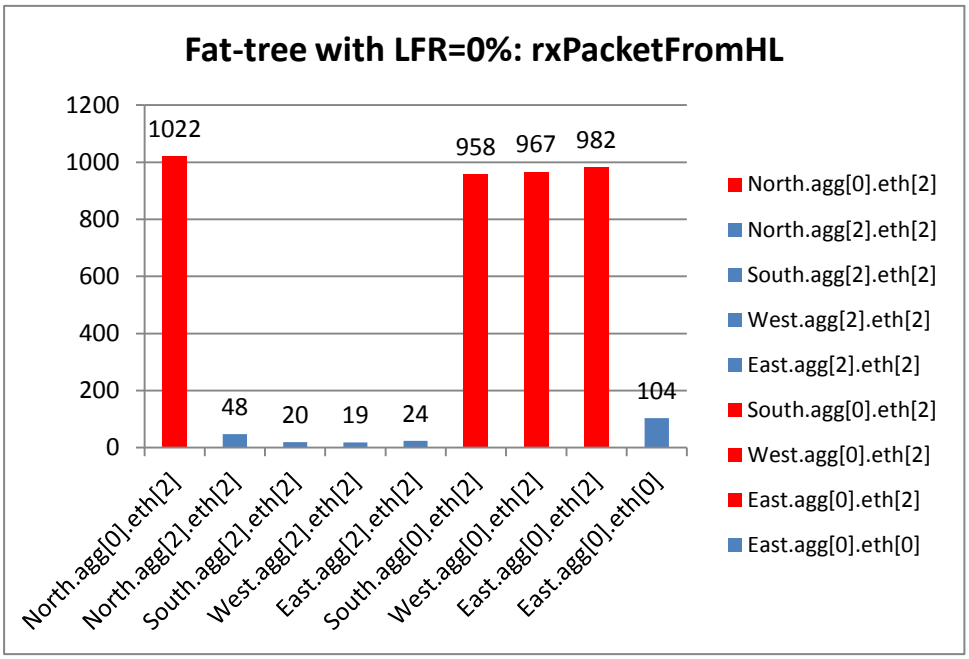


Figure 4.45 - Number of packet received from higher layer for Fat-tree with no failures

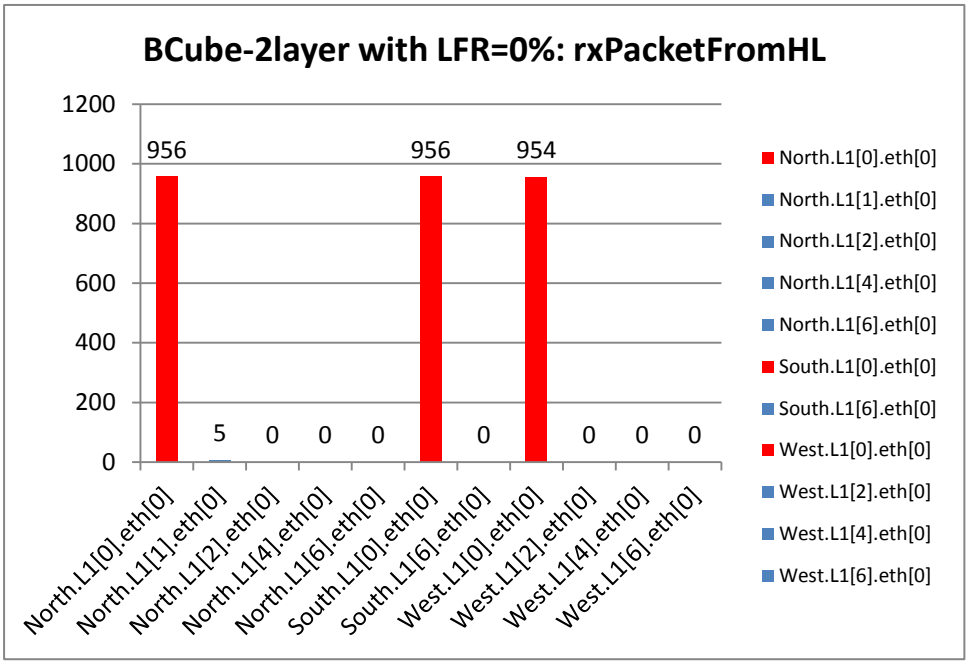


Figure 4.46 - Number of packet received from higher layer for BCube-2layer with no failures

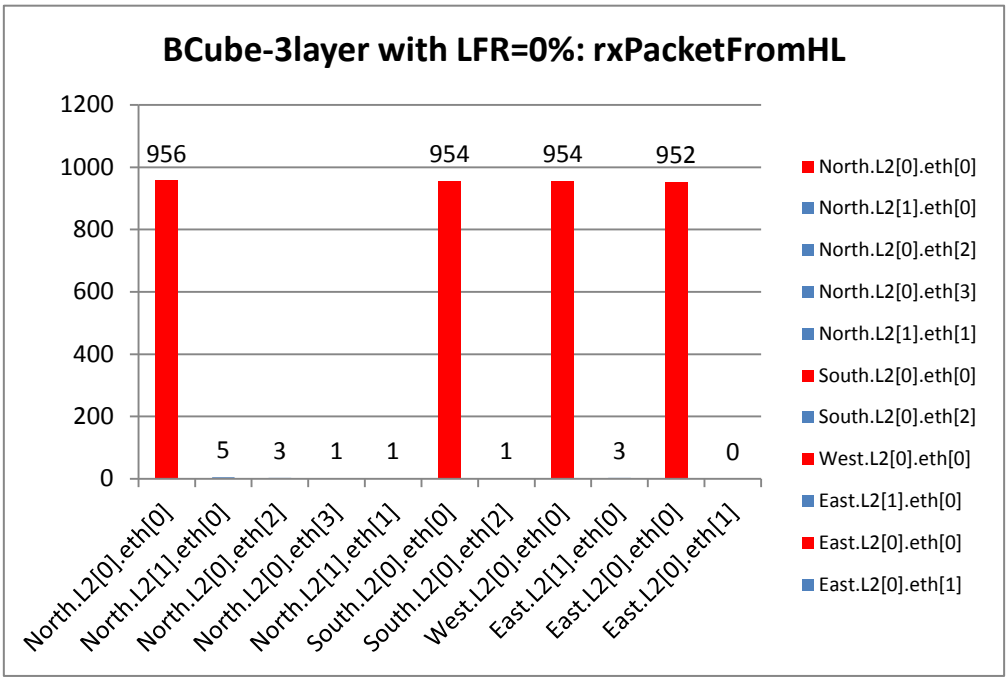


Figure 4.47 - Number of packet received from higher layer for BCube-3layer with no failures

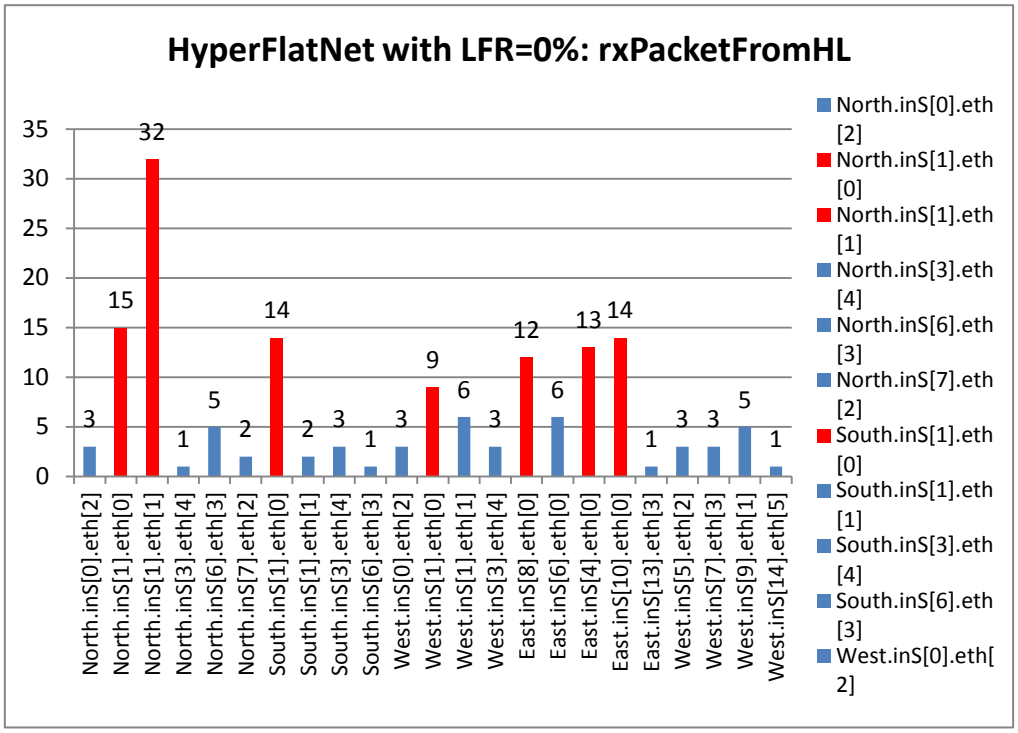


Figure 4.48 - Number of packet received from higher layer for HyperFlatNet with no failures

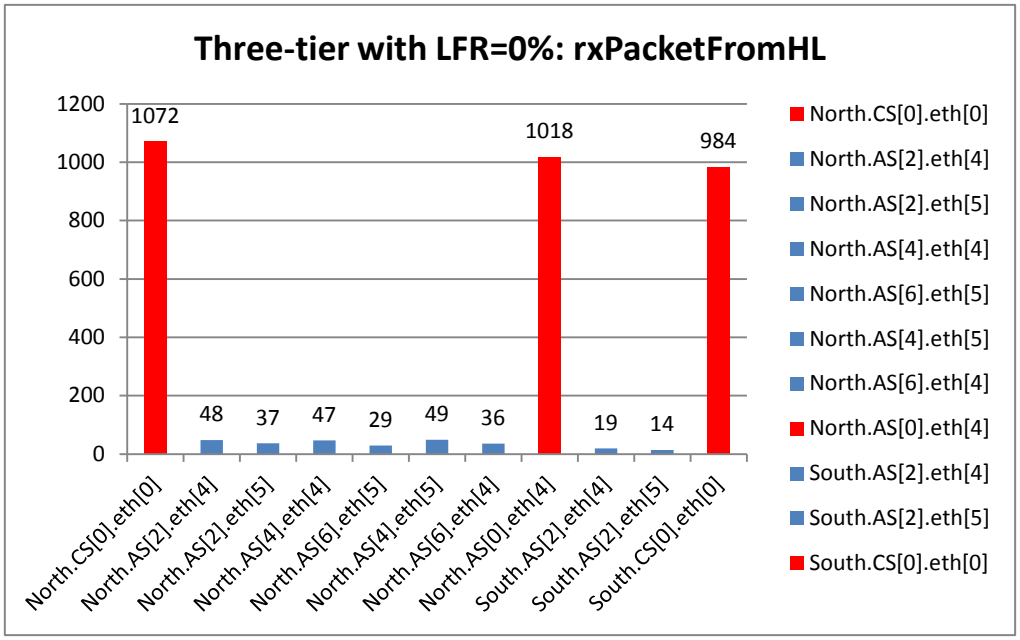


Figure 4.49 - Number of packet received from higher layer for Three-tier with no failures

Notation	Representation
F-SN	Failure of source node
F-DN	Failure of destination node
ASPL	Average Shortest Path Length
AWD	Average weighted degree (regardless of traffic load)
AND	Average nodal degree
AS[n]	Aggregation switches [index] in Fat-tree and Three-tier
L1[n]	Layer 2 switches [index] in BCube-2 layer
L2[n]	Layer 3 switches [index] in BCube-3 layer
Ins[n]	Internal Switches [index] in HyperFlatNet
CS[n]	Core switches [index] in Three-tier
S[n]	Server[index]

Table 4.20 - Notice on notation representation on table 4.21-25

Index	LFR	F-SN	F-DN	Diameter	Radius	ASPL	AWD	AND
Original	0%	-	-	11	6	7.441821725441855	16.231	2.329
1	0.260%	N.AS[0]	N.Edge[0]	12	6	7.522480051973147	16.173	2.323
2	0.521%	N.AS[2]	N.Edge[2]	12	6	7.60313837850444	16.115	2.317
3	0.781%	S.AS[2]	Edge[2]	13	7	7.683796705035731	16.058	2.311
4	1.042%	W.AS[2]	W.Edge[2]	13	7	7.7644550315670235	16	2.305
5	1.302%	E.AS[2]	E.Edge[2]	13	7	7.845113358098316	15.942	2.300
6	1.563%	S.AS[0]	S.Edge[0]	13	7	7.925771684629608	15.885	2.294
7	1.823%	W.AS[0]	E.Edge[0]	13	7	8.0064300111609	15.827	2.288
8	2.083%	E.AS[0]	E.Edge[0]	13	7	8.087088337692192	15.769	2.282
9	2.344%	E.AS[0]	E.CS[0]	15	8	8.262997451316819	15.712	2.277

Table 4.21 - Fat-tree network robustness metrics change according to increasing LFR

Index	LFR	F-SN	F-DN	Diameter	Radius	ASPL	AWD	AND
Original	0%	-	-	11	6	7.083401995788702	10.828	3.215
1	0.195%	N.L1[0]	N.S[0]	14	7	7.195166163141994	10.822	3.208
2	0.391%	N.L1[1]	N.S[1]	14	7	7.206408495834478	10.816	3.202
3	0.586%	N.L1[2]	N.S[2]	14	7	7.218236748146114	10.81	3.196
4	0.781%	N.L1[4]	N.S[4]	14	7	7.230650920076902	10.804	3.19
5	0.977%	N.L1[6]	N.S[6]	14	7	7.243651011626842	10.798	3.184
6	1.172%	S.L1[0]	S.S[0]	17	9	7.355415178980134	10.792	3.178
7	1.367%	S.L1[6]	S.S[6]	17	9	7.366657511672617	10.785	3.172
8	1.563%	W.L1[0]	W.S[0]	17	9	7.478421679025908	10.779	3.166
9	1.758%	W.L1[2]	W.S[2]	17	9	7.489664011718393	10.773	3.16
10	1.953%	W.L1[4]	W.S[4]	17	9	7.501492264030029	10.767	3.154
11	2.148%	W.L1[6]	W.S[6]	17	9	7.513906435960816	10.761	3.148

Table 4.22 - BCube-2 layers network robustness metrics change according to increasing LFR

Index	LFR	F-SN	F-DN	Diameter	Radius	ASPL	AWD	AND
Original	0%			17	9	10.181878154170928	37.657	3.091
1	0.195%	N.L2[0]	N.S[0]	17	9	10.382249782850106	37.611	3.087
2	0.391%	N.L2[1]	N.S[1]	18	9	10.398610240679046	37.564	3.082
3	0.586%	N.L2[0]	N.S[32]	18	9	10.456905366625985	37.518	3.077
4	0.781%	N.L2[0]	N.S[48]	18	9	10.515112533122231	37.471	3.073
5	0.977%	N.L2[1]	N.S17	18	9	10.550142384360809	37.424	3.068
6	1.172%	S.L2[0]	S.S[0]	18	9	10.75051401303999	37.377	3.063
7	1.367%	S.L2[0]	S.S[32]	18	9	10.808809138986927	37.33	3.059
8	1.563%	W.L2[0]	W.S[0]	18	9	11.009180767666106	37.283	3.054
9	1.758%	E.L2[1]	E.S[1]	18	9	11.01067607832789	37.237	3.049
10	1.953%	E.L2[0]	E.S[0]	19	10	11.225912854174226	37.19	3.044
11	2.148%	E.L2[0]	E.S[16]	22	11	12.061923453288035	37.143	3.04

Table 4.23 - BCube-3 layers network robustness metrics change according to increasing LFR

Index	LFR	F-SN	F-DN	Diameter	Radius	ASPL	AWD	AND
Original	0%	-	-	15	8	8.570738289532866	35.646	3.018
1	0.102%	N.ins[0]	N.S[0]	15	8	8.611051853755702	35.595	3.013
2	0.205%	N.ins[1]	N.ins[0]	17	9	8.856043179335604	35.544	3.008
3	0.307%	N.ins[1]	N.ins[2]	17	9	8.858703334832615	35.494	3.003
4	0.410%	N.ins[3]	N.S[2]	17	9	8.871567178564543	35.443	2.997
5	0.512%	N.ins[6]	N.S[63]	17	9	8.914412388357	35.392	2.992
6	0.615%	N.ins[7]	N.S[28]	17	9	8.916275782304183	35.342	2.987
7	0.717%	S.ins[1]	S.ins[0]	17	9	9.064499132557991	35.291	2.982
8	0.820%	S.ins[1]	S.ins[2]	17	9	9.067082182098567	35.241	2.977
9	0.922%	S.ins[3]	S.S[2]	17	9	9.0820792906252	35.19	2.972
10	1.025%	S.ins[6]	S.S[63]	17	9	9.100983100944548	35.139	2.967

…continued								
11	1.127%	W.ins[0]	W.S[0]	17	9	9.141296665167385	35.089	2.962
12	1.230%	W.ins[1]	N.ins[0]	19	10	9.386287990747284	35.038	2.957
13	1.332%	W.ins[1]	W.ins[2]	19	10	9.388948146244298	34.987	2.952
14	1.434%	W.ins[3]	W.S[2]	19	10	9.401811989976226	34.937	2.947
15	1.537%	E.ins[8]	E.ins[7]	19	10	9.404677761357064	34.886	2.942
16	1.639%	E.ins[6]	E.ins[5]	19	10	9.407492128766947	34.835	2.937
17	1.742%	E.ins[4	E.ins[3]	19	10	9.4104607080897	34.785	2.932
18	1.844%	E.ins[10]	E.ins[9]	19	10	9.41330077748506	34.734	2.927
19	1.947%	E.ins[13]	E.S[27]	19	10	9.433708153954893	34.684	2.922
20	2.049%	W.ins[5]	W.S[20]	19	10	9.439285484803701	34.633	2.916
21	2.152%	W.ins[7]	W.S[3]	19	10	9.458330656043179	34.582	2.911
22	2.254%	W.ins[9]	W.ins[10]	19	10	9.461209278416758	34.532	2.906
23	2.357%	W.ins[14]	W.S[53]	19	10	9.4627642485382	34.481	2.901

Table 4.24- HyperFlatNet network robustness metrics change according to increasing LFR

Index	LFR	F-SN	F-DN	Diameter	Radius	ASPL	AWD	AND
Original	0%	-	-	9	5	7.227265912611817	16.599	2.697
1	0.223%	N.CS[0]	N.AS[0]	9	5	7.236061368293049	16.542	2.692
2	0.446%	N.AS[2]	N.Edge[2]	9	5	7.23636121337309	16.536	2.686
3	0.669%	N.AS[2]	N.Edge[3]	9	5	7.236661058453133	16.53	2.68
4	0.893%	N.AS[4]	N.Edge[4]	9	5	7.236960903533174	16.524	2.674
5	1.116%	N.AS[6]	Edge[7]	9	5	7.237260748613217	16.519	2.669
6	1.339%	N.AS[4]	N.Edge[5]	9	5	7.237560593693258	16.513	2.663
7	1.562%	N.AS[6]	N.Edge[6]	9	5	7.237860438773301	16.507	2.657
8	1.786%	N.AS[0]	N.Edge[0]	9	5	7.238160283853342	16.501	2.651
9	2.009%	S.AS[2]	S.Edge[2]	9	5	7.238460128933385	16.496	2.646
10	2.232%	S.AS[2]	S.Edge[3]	9	5	7.238759974013426	16.438	2.64
11	2.455%	S.CS[0]	S.AS[0]	9	5	7.247555429694658	16.380	2.634

Table 4.25 - Three-tier network robustness metrics change according to increasing LFR

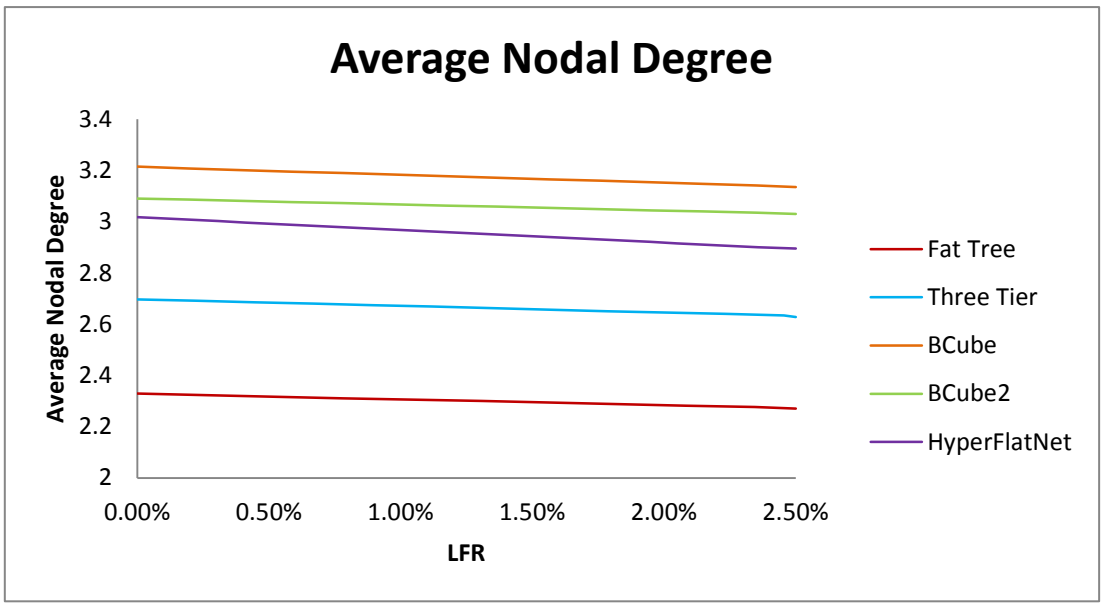


Figure 4.50 - Average Nodal Degree according to increasing LFR

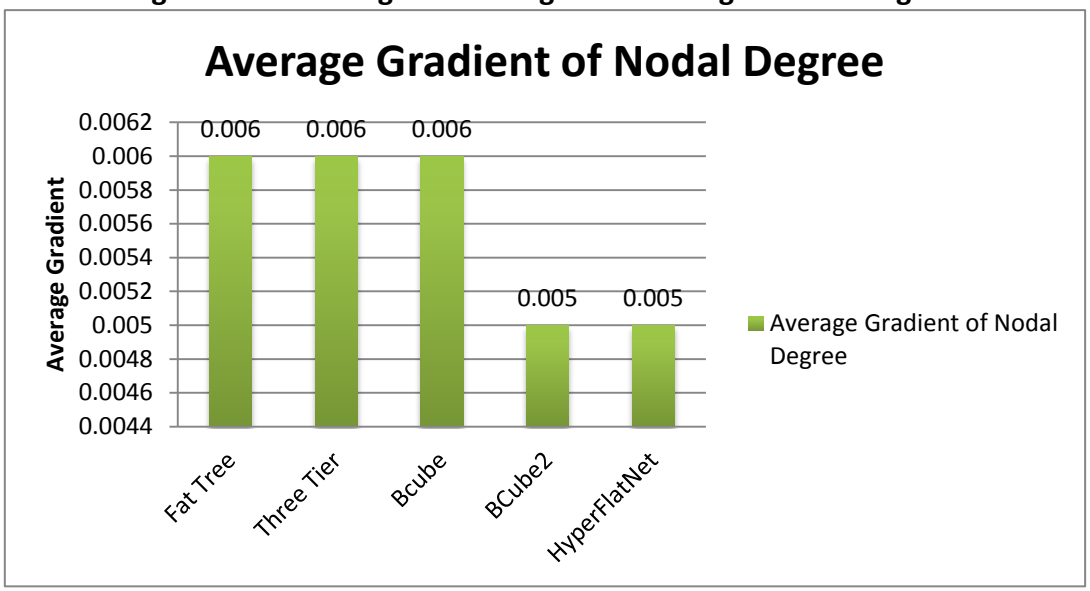


Figure 4.51 - Average gradient of nodal degree for various DCNs

Fat Tree	Three Tier	BCube	BCube2	HyperFlatNet
0.058	0.069	0.079	0.061	0.112

Table 4.26 – The gradient of average nodal degree

From table 4.26, we can see that by contrast to Fat-tree/Three-tier/BCube-2 layers and 3 layers, HyperFlatNet had a higher gradient of decreasing its average nodal degree, which implies that with the LFR increased, the HyperFlatNet mitigates the probability of connections being affected. In this case, even if the Fat-tree gained the lowest average nodal degree, it lacked the ability of "stopping" the connections that were being affected. On the other hand, several link failures may constitute one

effective connection failure between nodes, which is to say, the higher average degree of a node indicates a higher number of the neighbor edges of a node. This means it has a large failure ratio to make up for any connection failure to its neighboring nodes. Three server-centric DCN topologies gained the highest average nodal degree which means that BCube (2 layers and 3 layers), and HyperFlatNet are prone to be affected by link failure ratio increases.

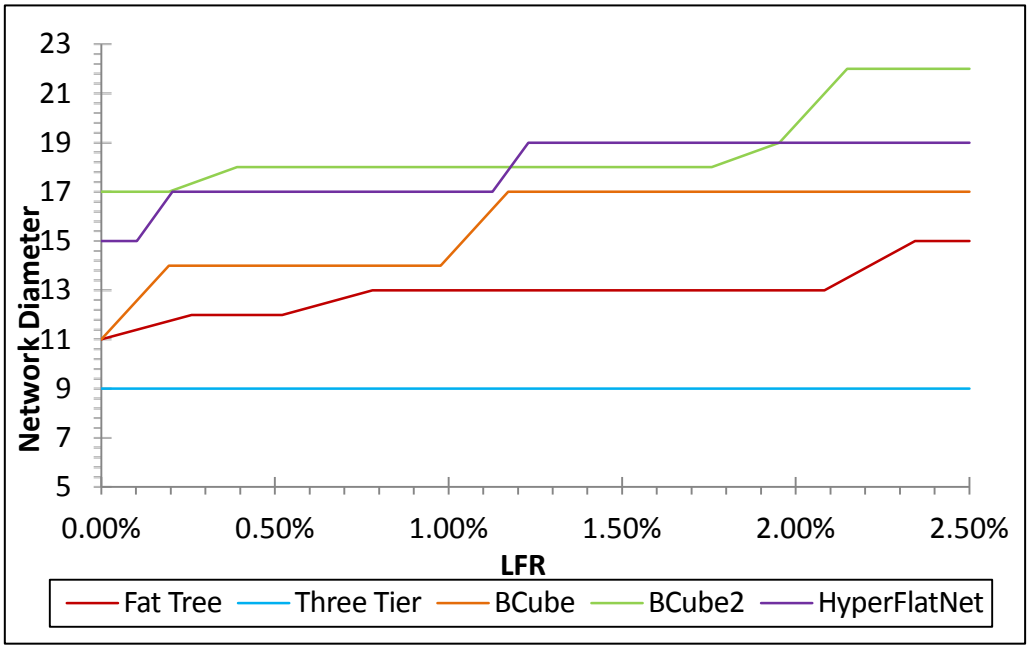


Figure 4.52 - Network Diameter for various DCNs according to increasing LFR

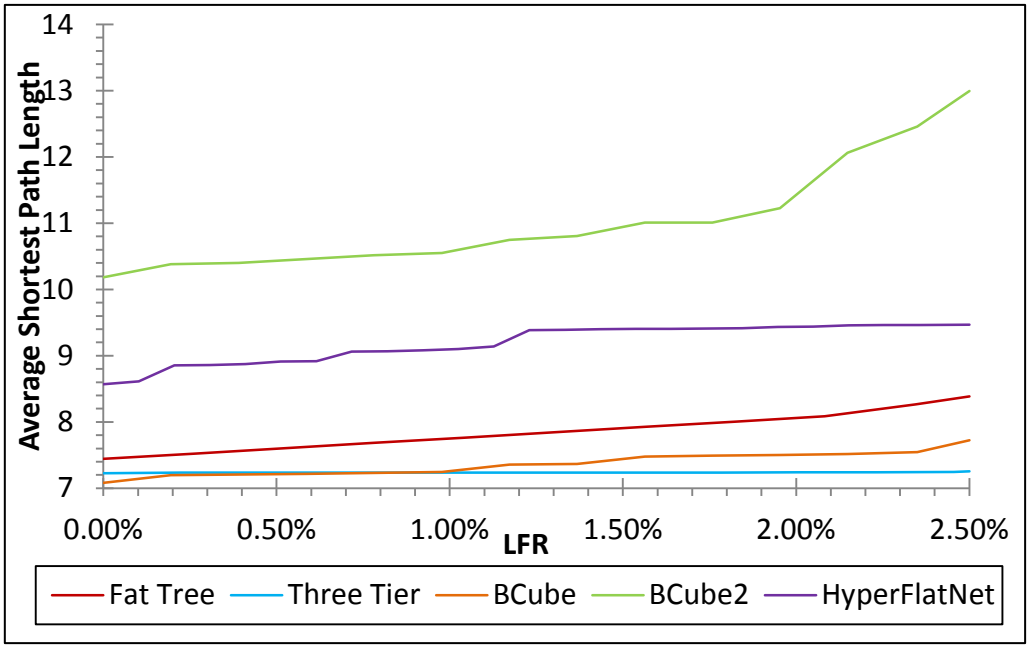


Figure 4.53 – Average Shortest Path Length for various DCNs according to increasing LFR

The network diameter is known as the maximum length among all the calculated shortest paths in a network. Under the all-to-all server traffic scenario, it is the longest path from one server to another reachable server. From a topological respect, a smaller network diameter indicates relatively lower network latency because of less opportunities of producing transmission delays and delays caused by the number of times information has to be queued. Ideally, that generates more effective routing.

From the figure shown above, the Three-tier DCN topology maintains the lowest network diameter with regard to the increase of LFR, which assumes that Three-tier DCN obtains the lowest network latency than all the others. On the other side, BCube with 3 layers is suspected to gain the highest network latency, as shown in Figure 4.53.

4.3.7 Case 2 - Scenario 2: Network Performance Metrics results analysis on Various DCNs according to increasing LFR

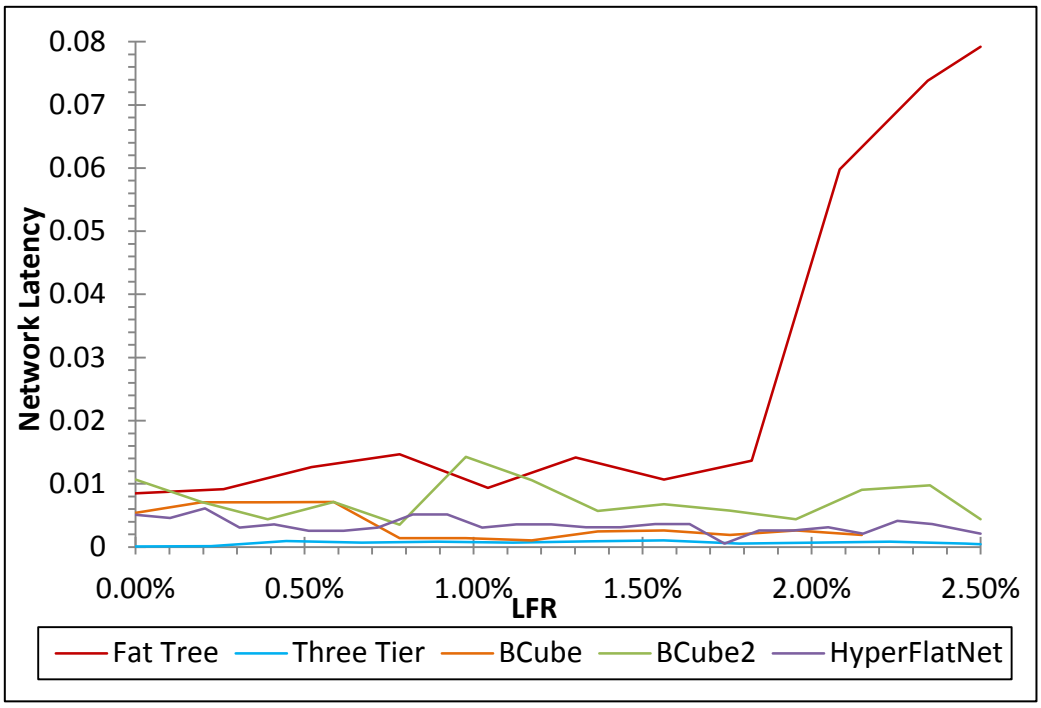


Figure 4.54 – Network Latency for various DCNs according to increasing LFR

As aforementioned, the three server-centric DCN architectures are predicted to have higher network latency because of the results of average nodal degree, network diameter and ASPL as shown in Figure 4.53. The previous assumption could have

been identified by the latency curves. BCube with 2 layers and 3 layers, and HyperFlatNet generated relative higher network latency, and the Three-tier showed the predicted result with the lowest network latency with the increase of LFR. This verified the hypothesis that it could take more chances to go through less transmission delay with a small ASPL and network diameter. However, BCube with 3 layers did not achieve the highest latency but this was achieved by Fat-tree. The discrepancy occurred mainly within the range 2.50% > LFR > 2.00%, as the traffic was generated as unbalanced. There was a coincidence that within that LFR range, the specific racks which generated a large number of data flows failed to transport packets to other racks on the shortest paths, so that more packets were necessarily sent on longer paths. Furthermore, the transmission time spent for a large proportion of total packets was increased as a result of the paths becoming longer while the packets received decreased due to longer queuing time. This exira time was required within the fixed UDP application transmitting time as shown in the following figure, which led to a steeper gradient after 2.00% LFR of average network latency of the Fat-tree.

The joggle of latency for HyperFlatNet was created by a random traffic load as the packet received stayed smooth with only a little decrease, which implies that the HyperFlatNet DCN topology is relatively more stable than the others. No matter how links were disconnected, the total number of packets received was maintained around 2000. According to the network latency model, the almost fixed packets received with the nearly unchanged latency result deduced that the delay of the packet *i* is relatively controlled within the acceptable time frame, so that the latency for HyperFlatNet presents a much more stable status. Also, from the respect of a topological view, HyperFlatNet possesses 1.9 times more links than BCube and 2.54 times more than the Fat-tree, and it contains the second largest number of switches which is supposed to have much more powerful stable transmission ability. Despite that, it has second largest ASPL and network diameter out of the five topologies.

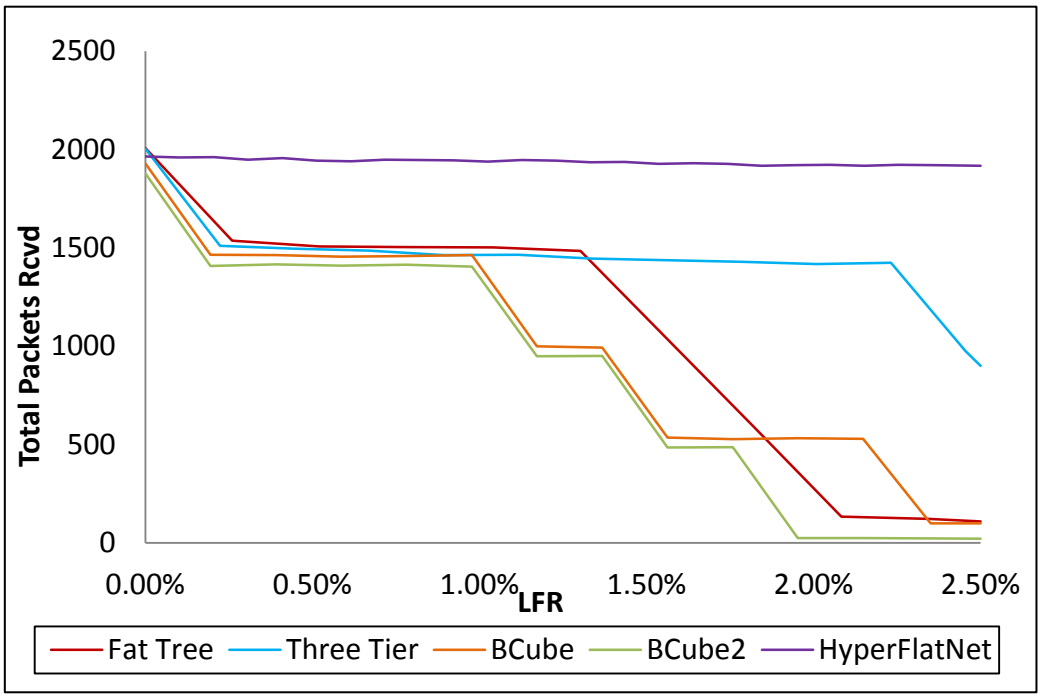


Figure 4.55 – Total Packets Received for various DCNs according to increasing LFR

There were a total of 2100 packets generated for all topologies. The packets received for all topologies presented as ladder-like trend except for HyperFlatNet, while the BCube-2 layers and 3 layers produced several steps downward after 1.00% LFR, and the Fat-tree got a sensitive response when LFR > 1.25% which appeared to be a lower reliability than the other tree-based topology, Three-tier. HyperFlatNet, with the largest number of links, shows the highest reliability against the other four topologies.

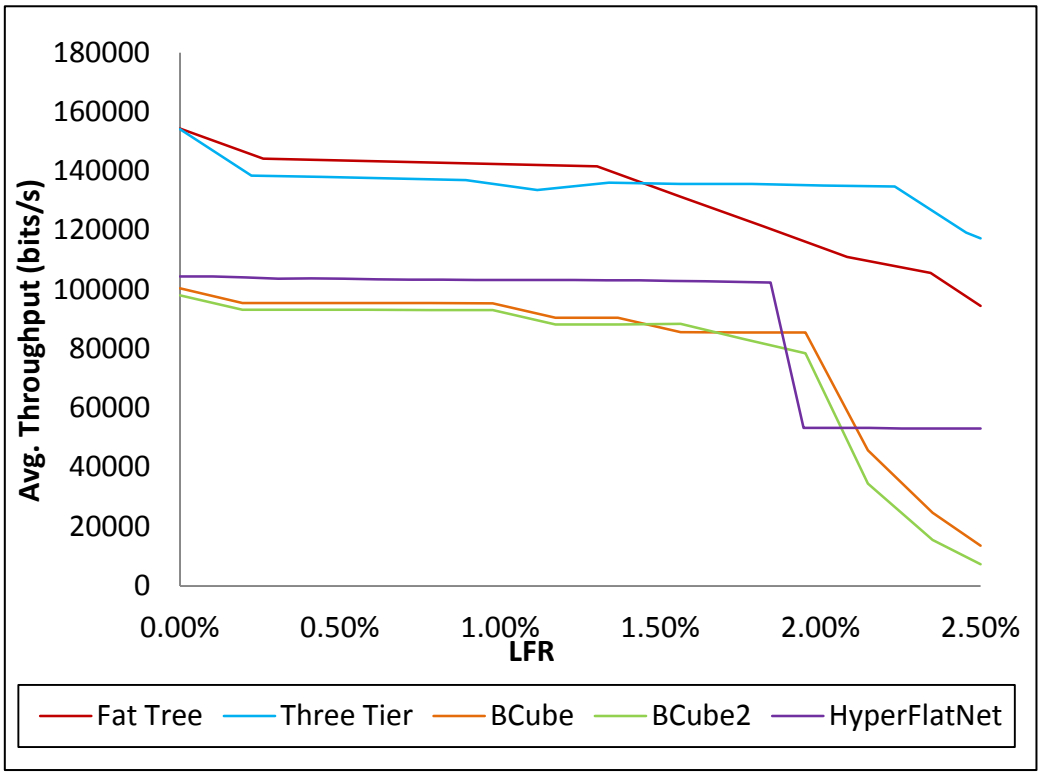


Figure 4.56 – Avg. Network Throughput for various DCNs according to increasing LFR

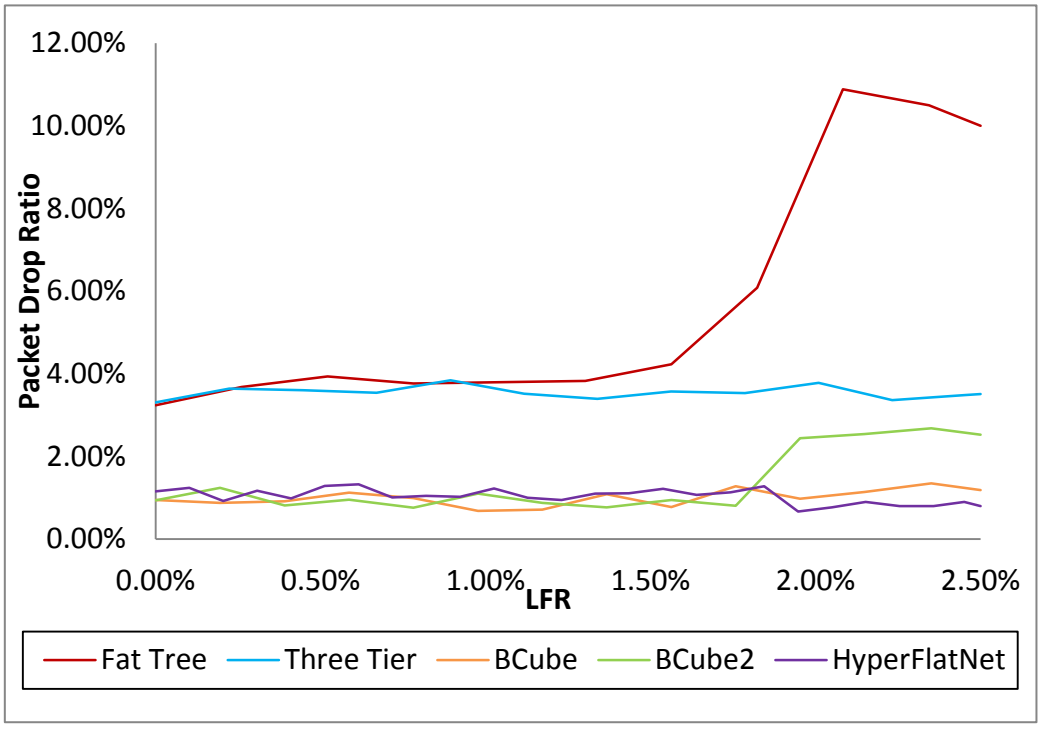


Figure 4.57 – Packet Drop Ratio for various DCNs according to increasing LFR

The above two figures show that the result curves could be clustered by different types of topology; the tree-based conventional Three-Tier and Fat-tree have very closed initial points on average throughput which is around 150,000 bits/s. Even

though the Fat-tree has less link oversubscription issues than Three-tier, with the link failure ratio increased, the Fat-tree decreases more dramatically than Three-Tier because Three-tier has more effective connections for maintaining the number of ASPLs, as well as for network robustness. As the traffic was defined as unbalanced, Figure 4.56 shows the source node ports of failed edges were used for receiving packets from a higher layer; as observed, the average network throughput decreased sharply after the 6^{th} link failure because the number of packets dropped dramatically at this moment.

The server-centric topologies all appeared steady until the LFR reached around 2.00%. Relatively, the HyperFlatNet was more robust within the limited LFR, but a fall occurred that indicates a specific effective connection failed. This may imply a substantial decrease of network performance. However, the performance could be still steady in a lower level. The BCube-2 layers and BCube-3 layers represent highly reliable capacity when LFR is less than 2.00%. The average throughput decreases approximate 20% which compared to Fat-tree at 31.25% and HyperFlatNet at 46.8%.

Similar with the average throughput, the result of PDR against LFR can also be classified into two clusters: tree-based DCN topologies and server-centric topologies. Obviously, the PDR for tree-based topologies are higher than the server-centric. The Fat-tree still presents an unstable trend; in addition to the decreasing number of shortest path length (SPL), the selected links for failure are prominent, as can be seen in Figure 4.56. Numerous effective connections failed to reliably relay messages after LFR moved beyond 1.50%, leading to an abnormal pattern on packets dropped. BCube-3 layers had more switches used than the other two server-centric topologies, and the failures on switches forced the packets to be transferred to alternative routes which caused the load burden on alternative switches. The drop tail queue mechanism compelled the packets to be dropped when the switch frame capacity achieved 100. This means the ratio for the packet dropping increased.

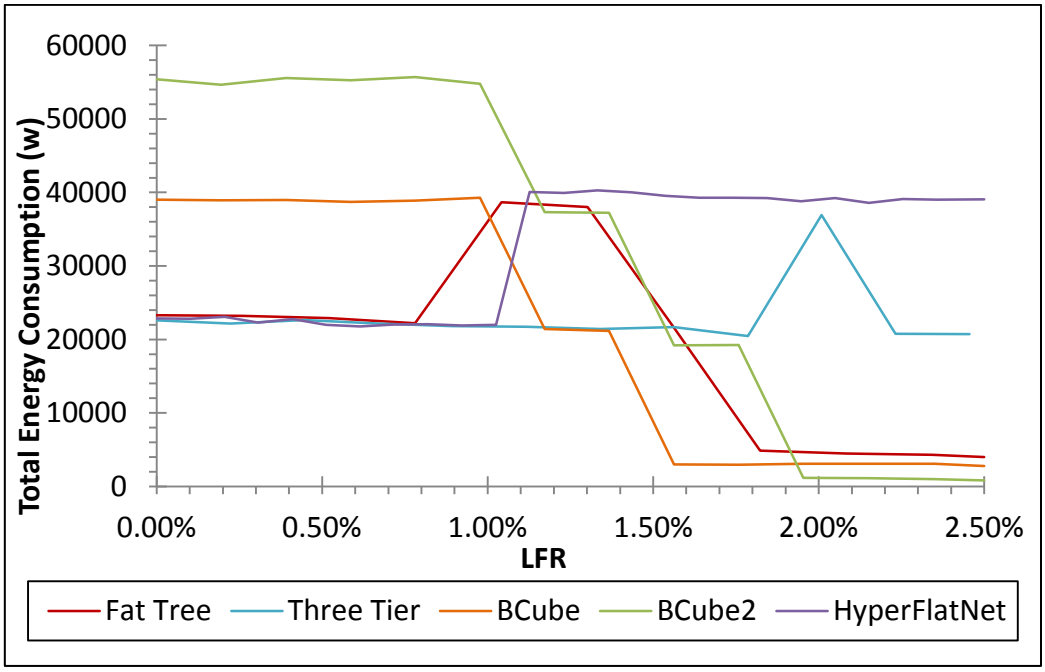


Figure 4.58 - Energy Consumption against increasing LFR

In this thesis, the researcher deployed the DVFS and DPM energy models simultaneously, and the energy consumption was adjusted according to the node voltage / CPU frequency by DVFS technique. Idle nodes were shut down automatically based on DPM technique so that energy consumption could be reduced to the lowest level. Within the limited simulation time, the energy consumption varied widely against the LFR, as shown in the above figure. In Fat-tree/Three-tier architectures, any failures of a top-of-rack (ToR) switch will cause the disconnection of a whole rack of servers, so that makes DVFS and DPM model dysfunction regionalized. This will produce a decrease in energy consumption. Otherwise, any failures of an end-of-rack (EoR) switch (aggregation or core switch) will push the switching load to be moved to other switches, thus the accumulated switching energy consumption will lead to the promotion of the curve. On the other hand, in order to capture the prominent topological feature, connections lie on important routes were selected as failure scenarios so that the results were amplified as representative. As BCube-2 layers and 3 layers results show, several link failures constitute a Cube fail, and the energy consumption decrease in these two topologies was due to the servers in that Cube not sending any packets, as shown in Figures 4.46 and 4.47. However, the number of packets received is determined by both switch load capacity and the degree

of failure, so an entire Cube cannot easily fail in BCube-3 layers because a Cube is supported by three layers of switches rather than BCube-2`s two layers of switches. The BCube-3 layers consumed more energy than two layers due to the higher number of switches which supported the workload concurrently. Comparatively, the HyperFlatNet is a more robust DCN topology so that there is less probability of link failures leading to server failure; only a few switches were disconnected during the LFR increasing process, so that the workload was easily transferred to the existing operating switches, which resulted in the energy increases.

Chapter 5 Conclusion and Future work

5.1 Conclusion

Cloud computing is a potential next generation technology in networking and telecommunication as it benefits daily life such as in the areas of global commerce, media and education. The cloud minimizes the cost of hardware and maintenance work from a commercial perspective, and there is no limitation on the availability of software. While datacenters play a very important role in cloud computing, the cost and the efficiency of datacenters are always the dominant topic in this cloud computing area; researchers have been evaluating datacenter efficiency with the aim of reducing cost and carbon dioxide output. This thesis focuses on such topics from a DCN topological aspect. However, because of the increasing use of DCN, QoS must be guaranteed in order to assure the normal operation, so DCN performance measurement is another important topic in this work. Therefore, the comparison among DCNs` architecture and performance (i.e., network throughput, packet latency, packet drop ratio, and number of total packets received) are regarded as the emphasis of this thesis. Five different popular types of DCN topology were compared from a topological view: Three-tier and Fat-tree, as well as the recursive MDC BCubes, and the high-performing and reliable network: HyperFlatNet. Moreover, fault tolerance is another network characteristic that indicates whether a network is robust or not. This thesis measured the network performance change versus an increasing link error rate to evaluate the fault tolerant ability of each DCN. For each DCN, network robustness metrics (i.e., average nodal degree, average weighted degree, average shortest path length, network diameter and radius) were evaluated and analyzed, centrality metrics (i.e., betweenness centrality, closeness centrality, eigenvector, and eccentricity) were evaluated for both nodes and edges. The network robustness metrics and the centrality

metrics determined the most critical nodes and edges, so that a number of links could be identified as the selection of failures. Such principle is valuable for cloud service providers to identify the network vulnerabilities, so that they are able to invest in the most critical network areas or hardware components with limited funds.

5.2 Main findings

DCNs performance is not merely a function of resource provisioning and allocation, but also it is a network-wide activity. This study revealed how the DCN QoS performance and robustness can be impacted on by the underlying network structure in a cloud environment. Some topological metrics were studied to reveal their impacts on DCN performances and robustness. Then the robustness of increasing LFR for different DCN architectures was evaluated, which also shows correlations to DCN performances. Under the same network settings, the topological metrics such as network diameter and ASPL can only be rough indicators of network latency because they cannot comprehensively explain the complicated DCN structure. The conventional Three-tier DCN in this case generated lowest average latency due to the amount of links deployed and its tree architecture preserved the highest efficiency on network transmission. Additionally, with the adoption of DVFS and fixed timeout DPM policy, the energy consumption by different DCN architectures illustrated irregular patterns as the failed links were selected to highlight the effects.

Q 1: Does the underlying DCN topology have an impact on the cloud network performance (QoS & energy)?

From the results of simulation studies, we can see that the different DCN topologies have various impacts on the cloud network performance (QoS and energy). For example, in case 2, under the same traffic scenario, the Fat-tree achieved the highest average network throughput, while the BCube-3 layers obtained the lowest. The Three-tier DCN topology got both the highest network latency and the packet drop ratio. On the other hand, BCube-3 layers consumed the largest energy compared

to all the other DCN topologies. The DCN architecture produces the difference in the network performance and energy consumption.

Q 2: What network topological metrics can be used as an index to quantify the cloud data center performance in the cases of energy efficiency and QoS?

It is difficult to find one accurate metric for quantifying data center network performance and also energy efficiency, but it is able to monitor the performance on the basis of comprehensively analyzed network robustness metrics and centrality metrics according to the traffic load pattern where the faults happened. For example, the failure of a critical link has much significant impact on the DCN performance compared to less critical links. So the foremost thing is to determine the critical links. Therefore, the method in this thesis was to use centrality metrics (betweenness, closeness, eigenvector, eccentricity) coupled with analyzing traffic flow patterns to determine the critical links.

Q 3: How to use this metric to determine the critical nodes/links in a data center network?

For example, aggregation switches were found to be the most important clusters of important nodes. And if the aggregation switch [0] got the heaviest workload, then AS[0] became the most critical node in the DCN. The failure of AS[0] decreased the network performance by the largest proportion (throughput, packet drop rate), and latency was lowered due to queuing time decreasing when a large number of packets were dropped. Energy consumption showed an irregular pattern which should be considered case-by-case.

5.3 Future work

Considering the work covered in this thesis occurred within a limited time period, it would be useful to highlight some areas to be further investigated.

Firstly, more realistic and complex DCN energy aware performance models will be further developed by considering various network scenarios, and also a better topological indicator needs to be sought.

Secondly, a larger number of experiments should be conducted in the next research stage to minimize the average error. The cases are also needed to be improved; link error rate should be expanded so that more links are selected to be failed. Furthermore, failed links could be selected randomly to cater to the realistic scenarios.

Glossary

ASPL: Average Shortest Path Length

AWS: Amazon Web Services

BWs: Balanced Workloads

CDN: Content Distribution Network

CIWs: Computationally Intensive Workloads

DC: Data center

DCN: Data Center Network

DIWs: Data-intensive Workloads

DPM: Dynamic Power Management

DVFS: Dynamic Voltage/Frequency Scaling

HDT: high data transferring

HPC: High-Performance Computing

IaaS: Infrastructure as a Service

ICT: Information Communications Technologies

LCM: Linked Clusters Maximization

LDT: low data transferring

LFR: Link Failure Rate

MDC: Modular Data Center

NIST: National Institute of Standard and technology

NPE: Network Performance Evaluation

OPEX: Operational Expenses

PaaS: Platform as a Service

PCL: Prediction Confidence Level

QoS: Quality of Service

RDS: Relational Database Service

SaaS: Software as a Service

SLAs: Service Level Agreements

SLB: Server load balancing

SOA: Service-oriented Architecture

SOAP: Simple Object Access Protocol

TCO: Total Cost of Ownership

VBB/VBC: Volume Based Billing/Control

Reference

[1] S. Khan and A. Zomaya, *Handbook on data centers*.

[2] Y. Liu, J. Muppla, M. Veeraghavan, D. Lin and M. Hamdi, *Data Center Networks*. Springer International Publishing, 2013, pp. 1-5.

[3] P. Bailis and K. Kingsbury, "The network is reliable", *Communications of the ACM*, vol. 57, no. 9, pp. 48-55, 2014.

[4] Amazon, "Random network interruptions with my EC2 instances, failover impossible", 2013. [Online]. Available: https://forums.aws.amazon.com/thread.jspa?messageID=454155. [Accessed: 19- Sep- 2015].

[5] GitHub, "Downtime last Saturday", 2012. [Online]. Available: https://github.com/blog/1364-downtime-last-saturday. [Accessed: 28- Sep- 2015].

[6] K. Choo, 'Mobile Cloud Storage Users', *IEEE Cloud Computing*, vol. 1, no. 3, pp. 20-23, 2014.

[7] A. Beitelmal and C. Patel, "Thermo-Fluids Provisioning of a High Performance High Density Data Center", *Distrib Parallel Databases*, vol. 21, no. 2-3, pp. 227-238, 2006..

[8] Gartner.com, 'Technology Research | Gartner Inc., 2015. [Online].

[9] A. Banerjee, T. Mukherjee, G. Varsamopoulos and S. Gupta, 'Integrating cooling awareness with thermal aware workload placement for HPC data centers', *Sustainable Computing: Informatics and Systems*,

[10] J. Carter and K. Rajamani, 'Designing Energy-Efficient Servers and Data Centers', *Computer*, vol. 43, no. 7, pp. 76-78, 2010.

[11] D. Kliazovich, P. Bouvry and S. Khan, 'GreenCloud: a packet-level simulator of energy-aware cloud computing data centers', *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263-1283, 2010.

[12] R. Couto, S. Secci, M. Campista and L. Costa, "Reliability and Survivability Analysis of Data Center Network Topologies", *J Netw Syst Manage*, 2015.

[13] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. Malluhi, N. Tziritas, A. Vishnu, S. Khan and A. Zomaya, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems", *Computing*, 2014.

[14] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in Data Center Networks", *Computer Communications*, vol. 40, pp. 1-21, 2014.

[15] W. Lloyd, S. Pallickara, O. David, M. Arabi, T. Wible and J. Ditty, "Demystifying the Clouds: Harnessing Resource Utilization Models for Cost Effective Infrastructure Alternatives", *IEEE Transactions on Cloud Computing*, pp. 1-1, 2015.

[16] Costa, P., Migliavacca, M., Pietzuch, P., & Wolf, A. L. "NaaS: Network-as-a-Service in the Cloud." In *Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE* (Vol. 12, pp. 1-1).

[17] P. Mell and T. Grance, "The NIST Definition of Cloud Computing.", *Communications of the ACM.*, vol. 53, no. 6, pp. p50-50. 2/3p., 2010.

[18] C. Liu, *Service-oriented computing*. Heidelberg: Springer, 2012.

[19] L. Wang, D. Chen, J. Zhao and J. Tao, "Resource management of distributed virtual machines", *IJAHUC*, vol. 10, no. 2, p. 96, 2012.

[20] L. Wang, D. Chen and F. Huang, "Virtual workflow system for distributed collaborative scientific applications on Grids", *Computers & Electrical Engineering*, vol. 37, no. 3, pp. 300-310, 2011.

[21] L. Wang, G. von Laszewski, D. Chen, J. Tao and M. Kunze, "Provide Virtual Machine Information for Grid Computing", *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 6, pp. 1362-1374, 2010.

[22] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia (2010) "A view of cloud computing." Commun ACM 53(4):50–58

[23] W. Jansen (2011) “Cloud hooks: security and privacy issues in cloud computing.” In: 44th Hawaii international conference on systems science (HICSS), pp 1–10.

[24] N. Sadashiv, S. Kumar (2011) “Cluster, grid and cloud computing: a detailed comparison.” In: 6th international conference on computer science and education (ICCSE 2011), pp 477–482

[25] L. Barroso, U.Hölzle(2009) The datacenter as a computer: an introduction to the design ofwarehousescale machines, 1st edn. In: Hill MD (ed) Morgan and Claypool Publishers, University of Wisconsin, Madison

[26] A. Berl, E. Gelenbe, M. Girolamo, G. Giuliani, H. Meer, M. Dang, K. Pentikousis (2010) “Energyefficient cloud computing.” Comput J 53(7):1045–1051

[27] J. Kaplan, W. Forrest and N. Kindler, "Revolutionizing Data Center Energy Efficiency", McKinsey & Company, 2016.

[28] S.Srikantaiah, A. Kansal, F. Zhao (2008) “Energy aware consolidation for cloud computing.” In: Conference on power aware computer and systems (HotPower ’08)

[29] Y. Lee, A. Zomaya (2012) “Energy efficient utilization of resources in cloud computing systems.” J Supercomput 60(2):268–280. doi:10.1007/s11227-010-0421-3

[30] J. Paradiso, T. Starner (2005) “Energy scavenging formobile andwireless electronics.” Pervasive Comput 4(1):18–27

[31] M. Elnozahy, M. Kistler, R. Rajamony (2002) “Energy-efficient server clusters.” Power aware computer systems, vol 2325. Springer, Berlin, pp 179–197

[32] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron (2003) “Power-aware QoS management in web servers.” In: Real-time systems symposium (RTSS 2003), pp 63–72

[33] T. Horvath, T. Abdelzaher, K. Skadron, X. Liu (2007) “Dynamic voltage scaling in multitier web servers with end-to-end delay control.” IEEE Trans Comput 56(4):444–458

[34] X. Liu, P. Shenoy, W. Gong (2004) “A time series-based approach for power management in mobile processors and disks.” In: 14th international workshop on network and operating systems support for digital audio and video (NOSSDAV ’04), pp 74–79

[35] D. Steere, A. Goel, J. Gruenberg, D. Mcnamee, C. Pu, L. Walpole (1999) “A feedback-driven proportion allocator for real-rate scheduling.” In: Third symposium on operating system designand implementation (OSDI), pp 145–158

[36] Chang Ge, Zhili Sun and Ning Wang, "A Survey of Power-Saving Techniques on Data Centers and Content Delivery Networks", *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1334-1354, 2013.

[37] F. Fang and X. Yu, "Design and Implementation of Next-Generation Data Center Infrastructure", *AMM*, vol. 513-517, pp. 1316-1319, 2014.

[38] Al-Fares, M., Loukissas, A., & Vahdat, A. (2008, August). A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review* (Vol. 38, No. 4, pp. 63-74). ACM.

[39] Google Whitepaper (2011) Google’s green data centers: network POP case study. Google. http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en/us/corporate/datacenter/dc-best-practices-google.pdf

[40] N. Gorti, "Application aware performance, power consumption, and reliability tradeoff" (2014). Graduate Theses and Dissertations. Paper 13933.", Ph.D, Iowa State University, 2016. [41] IBM. Ibm watson, 2013.

[41] P. Bosshart, C. Hewes, Mi-Chang Chang, Kwok-Kit Chau, C. Hoac, T. Houston, V. Kalyan, S. Lusky, S. Mahant-Shetti, D. Matzke, K. Ruparel, Ching-Hao Shaw, T. Sridhar and D. Stark, "A 553K-transistor LISP processor chip", *1987 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*.

[42] "Efficiency, Power, Cores... | TOP500 Supercomputer Sites", *Top500.org*, 2016. [Online]. Available: http://www.top500.org/statistics/efficiency-power-cores/. [Accessed: 20- May- 2015].

[43] M. Dayarathna, Y. Wen and R. Fan, "Data Center Energy Consumption Modeling : A Survey", *IEEE Communications Surveys & Tutorials*, pp. 1-1, 2015.

[44] J. Koomey, C. Belady, M. Patterson, A. Santos and K. Lange, *Assessing Trends Over Time in Performance, Costs, and Energy Use for Servers*, 1st ed. 2009.

[45] A. Howard and J. Holmes, "Addressing data center efficiency: lessons learned from process evaluations of utility energy efficiency programs", *Energy Efficiency*, vol. 5, no. 1, pp. 137-148, 2011.

[46] G. Baccarani, M. Wordeman, and R. Dennard, “Generalized scaling theory and its application to a ¼ micrometer MOSFET design,” IEEE Trans. Electron Devices IEEE Transactions on Electron Devices, vol. 31, no. 4, pp. 452–462, 1984.

[47] Intel. Dynamic data center power management: Trends, issues, and solutions

[48] Ram Viswanath, Vijay Wakharkar, Abhay Watwe, Vassou Lebonheur, et al. Thermal performance challenges from silicon to systems. 2000.

[49] Roger Schmidt. Challenges in electronic cooling opportunities for enhanced thermal management techniquesmicroprocessor liquid cooled minichannel heat sink. *Heat Transfer Engineering*, 25(3):3{12, 2004.

[50] Ravi Mahajan and Chia-pin Chiu. Cooling a microprocessor chip. *Proceedings of the IEEE*, 94(8), 2006.

[51] L. Shang, L. Peh, and N. Jha, “Power-efficient Interconnection Networks: Dynamic Voltage Scaling with Links,” IEEE Comput. Arch. Lett. IEEE Computer Architecture Letters, vol. 1, no. 1, pp. 6–6, 2002.

[52] M. Al-Fares, A. Loukissas and A. Vahdat, 'A scalable, commodity data center network architecture', SIGCOMM Comput. Commun. Rev., vol. 38, no. 4, pp. 63-74, 2008.

[53] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang and S. Lu, 'BCube', *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, p. 63, 2009.

[54] M. Brocanelli, W. Zheng and X. Wang, "Reducing the expenses of geo-distributed data centers with portable containerized modules", *Performance Evaluation*, vol. 79, pp. 104-119, 2014.

[55] Z. Chkirbene, S. Foufou, M. Hamdi and R. Hamila, "HyperFlatnet: A Novel Network Architecture For Data Centers", in *Communication Workshop (ICCW), 2015 IEEE International Conference, London*, 2015, pp. 1877 - 1882.

[56] C. Liu, A. Kind and T. Liu, "Summarizing Data Center Network Traffic by Partitioned Conservative Update", *IEEE Communications Letters*, vol. 17, no. 11, pp. 2168-2171, 2013.

[57] Cisco, "Global Cloud Index (GCI)", 2014. [Online]. Available: http://www.cisco.com/c/en/us/solutions/service-provider/global-cloud-index-gci/index.html. [Accessed: 28- Oct- 2015].

[58] K. Moiseev, A. Kolodny and S. Wimer, "Timing-aware power-optimal ordering of signals", *ACM Transactions on Design Automation of Electronic Systems*, vol. 13, no. 4, pp. 1-17, 2008.

[59] "Digital integrated circuits: analysis and design", *Choice Reviews Online*, vol. 42, no. 01, pp. 42-0323-42-0323, 2004.

[60] S. Djosic and M. Jevtic, "Dynamic voltage and frequency scaling algorithm for fault-tolerant real-time systems", *Microelectronics Reliability*, vol. 53, no. 7, pp. 1036-1042, 2013.

[61] "Global Provider of Innovative Graphics, Processors and Media Solutions | AMD", *Amd.com*, 2016. [Online]. Available: http://www.amd.com. [Accessed: 25- Aug- 2015].

[62] "Resource & Design Center", *Intel*, 2016. [Online]. Available: http://www.intel.com/content/www/us/en/design/resource-design-center.html. [Accessed: 16- Aug- 2015]..

[63] T. Okuma, H. Yasuura, and T. Ishihara. Software energy reduction techniques for variable voltage processors. *IEEE Design Test of Computers*, 18(2):31–41, Mar. 2001.

[64] P. Langen and B. Juurlink. Leakage-aware multiprocessor scheduling. *J. Signal Process. Syst.*,57(1):73–88, 2009.

[65] G. Chen, K. Malkowski, M. Kandemir, and P. Raghavan. Reducing power with performance constraints for parallel sparse applications. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, page 8 pp., Apr. 2005.

[66] Wang, L., Khan, S.U., Chen, D., Kolodziej, J., Ranjan, R., Xu, C.Z., Zomaya, A.Y.: Energy aware parallel task scheduling in a cluster. Future Generation Comp. Syst **29**(7) (2013) 1661–1670

[67] D. Shin and J. Kim, "Power-aware scheduling of conditional task graphs in real-time multiprocessor systems," *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 408–413, 2003.

[68] L. Benini and G. De Micheli, Dynamic Power Management:Design Techniques and CAD Tools, Kluwer Academic Publishers, 1997.

[69] Chen G, He W, Liu J, Nath S, Rigas L, Xiao L, Zhao F (2008) Energy-aware server provisioning and load dispatching for connection-intensive internet services. In: The 5th USENIX symposium on networked systems design and implementation, Berkeley, CA, USA

[70] Fan X, Weber W-D, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In:Proceedings of the 34th annual international symposium on computer architecture (ISCA '07). ACM, New York, pp 13–23.

[71] Yung-Hsiang Lu , Giovanni De Micheli, Comparing System-Level Power Management Policies, IEEE Design & Test, v.18 n.2, p.10-19, March 2001.

[72] A. Karlin, M. Manesse, L. McGeoch and S. Owicki, "Competitive Randomized Algorithms for Nonuniform Problems", Algorithmica, pp. 542-571, 1994.

[73] F. Douglis, P. Krishnan, and B. Bershad. Adaptive Disk Spin-Down Policies for Mobile Computers. In Computing Systems, volume 8, pages 381–413, 1995.

[74] C.-H. Hwang and A. C. Wu. A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation. In International Conference on Computer-Aided Design, pages 28–32, 1997.

[75] E.-Y. Chung, L. Benini, and G. D. Micheli. Dynamic power management using adaptive learning tree. In International Conference on Computer-Aided Design, pages 274–279, 1999.

[76] Y.-H. Lu and G. D. Micheli. Adaptive Hard Disk Power Management on Personal Computers. In Great Lakes Symposium on VLSI, pages 50–53, 1999.

[77] Fog Creek Software. 2012. May 5-6 network maintenance post-mortem; http://status.fogcreek.com/2012/05/may-5-6-network-maintenance-post-mortem.html.

[78] Amazon Web Services. 2011. Summary of the Amazon EC2 and Amazon RDS service disruption in the U.S. East region; http://aws.amazon.com/message/65648/.

[79] Omnetpp.org, "OMNeT++ Discrete Event Simulator - Home", 2015. [Online]. Available: https://omnetpp.org/. [Accessed: 13- Apr- 2015].

[80] A. Varga, "OMNeT++ - Manual", *Omnetpp.org*, 2016. [Online]. Available: https://omnetpp.org/doc/omnetpp/manual/usman.html#sec101. [Accessed: 23- Aug- 2015].

[81] Cloudnetsim.seecs.edu.pk, 2016. [Online]. Available: http://cloudnetsim.seecs.edu.pk/. [Accessed: 10- May- 2015].

[82] A. W. Malik, K. Bilal, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, R. Buyya, 'CloudNetSim++: A toolkit for Data Center Simulations in OMNeT++', *Proceedings of the 11th International Conference on High-capacity Optical Networks and Enabling/ Emerging Technologies, Dec*, 2014, Charlotte United States.

[83] A. Malik, "CloudNetSim++: A Toolkit for Data Center Simulations in OMNET++", 2015.

[84] S. Heymann, “Gephi,” Encyclopedia of Social Network Analysis and Mining, pp. 612–625, 2014.

[85] P. Desmedt, "Prix", *usinenouvelle.com*, 2011. [Online]. Available: http://www.usinenouvelle.com/article/prix-science-sebastien-heymann-le-cartographe-des-donnees.N164939. [Accessed: 16- Sep- 2015].

[86] Gephi.org, "Gephi - The Open Graph Viz Platform", 2016. [Online]. Available: https://gephi.org/. [Accessed: 12- Sep- 2015].

[87] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services.", 2008, pp. 337–350.

[88] Li Shang, L. Peh and N. Jha, 'Power-efficient Interconnection Networks: Dynamic Voltage Scaling with Links', *IEEE Computer Architecture Letters*, vol. 1, no. 1, pp. 6-6, 2002.

[89] "Intel® Xeon® Processor E5 Family", *Intel*, 2016. [Online]. Available: http://www.intel.com/content/www/us/en/processors/xeon/xeon-processor-e5-family.html. [Accessed: 16- Jul- 2015].

[90] R. Goonatilake and R. Bachnak, 'Modeling Latency in a Network Distribution', *Network and Communication Technologies*, vol.1, 2012.

[91] Jayanthi, 'Improving the performance and reducing bit error rate on wireless deep fading environment receivers', *Journal of Computer Science*, vol. 10, no. 3, pp. 458-468, 2014.

[92] R. Diestel, *Graph theory*. New York: Springer, 1997.

[93] G. McCusker, "A Graph Model for Imperative Computation", *Logical Methods in Computer Science*, vol. 6, no. 1, 2010.

[94] B. Wu, K. Yeung and P. Ho, "Virtual Topology Design for Minimizing Network Diameter and Average Hop Count in WDM Networks", *Journal of Optical Communications and Networking*, vol. 2, no. 12, p. 1077, 2010.

[95] Y. Qin, "Computer Network Attack Modeling and Network Attack Graph Study", *AMR*, vol. 1079-1080, pp. 816-819, 2014.

[96] L. Freeman, "A Set of Measures of Centrality Based on Betweenness", *Sociometry*, vol. 40, no. 1, p. 35, 1977.

[97] "A GraphML-based Visualization Framework for Workflow-Performers' Closeness Centrality Measurements", *KSII TIIS*, vol. 9, no. 8, pp. 3216-3230, 2015.

[98] Øystein Ore, Theory of graphs [3rd ed., 1967], Colloquium Publications, American Mathematical Society, p. 104

[99] "Feature Column from the AMS", *American Mathematical Society*, 2016. [Online]. Available: http://www.ams.org/samplings/feature-column/fcarc-pagerank. [Accessed: 15- Aug- 2015].

[100] M. E. J. Newman. "The mathematics of networks" (PDF). Retrieved 2006-11-09.

[101] G. Consortium, "Gephi: Exploratory networks analysis software", *Slideshare.net*, 2010. [Online]. Available: http://www.slideshare.net/gephi/gephi-exploratory-networks-analysis-software-4931589. [Accessed: 21- Sep- 2015].

Appendix A: CloudNetSim++ environment

Sample C++ codes for Three-tier DCN topology implementation in CloudNetSim++

A 64-server Three-tier DCN can be modelled as the following code on the basis of CloudNetSim++ simulator.

```
module Racks_Three64
{
    parameters:
        int N @prompt("Number of servers per Rack");//inputs N=8
        @display("bgb=506,467");
    gates:
        inout iogate[];
    submodules:
        computingServer[N]: myComputingNode {
            @display("p=120,305,m,6,80,80");
        }
        edgeRouter: RouterGreenCloud {
            @display("p=202,148");
        }
    connections:
        for i=0..N-1 {
            edgeRouter.ethg++ <--> Eth1G{per=0.008;} <--> computingServer[i].ethg++;
        }
        edgeRouter.ethg++ <--> iogate++;
        edgeRouter.ethg++ <--> iogate++;
}
module MeshCDatacenter_Three64
{
    parameters:
        int aggregateswitch = default(8);
        int corerouter = default(4);
        int accessswitch = default(8);
        @display("bgb=954,456");
    gates:
```

```
        inout iogate[];
    submodules:
        AS1[aggregateswitch]: RouterGreenCloud
        CS1[corerouter]: RouterGreenCloud
        Rks[accessswitch]: Racks_Three64
    connections allowunconnected:
        for i=0..corerouter-1, for j=0..aggregateswitch-1 {CS1[i].ethg++ <-->
Eth10G {per=0.008;} <--> AS1[j].ethg++;}
        CS1[0].ethg++ <--> Eth10G <--> iogate++;
        Rks[0].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[0].ethg++;
        Rks[0].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[1].ethg++;
        Rks[1].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[0].ethg++;
        Rks[1].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[1].ethg++;
        Rks[2].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[2].ethg++;
        Rks[2].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[3].ethg++;
        Rks[3].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[2].ethg++;
        Rks[3].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[3].ethg++;
        Rks[4].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[4].ethg++;
        Rks[4].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[5].ethg++;
        Rks[5].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[4].ethg++;
        Rks[5].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[5].ethg++;
        Rks[6].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[6].ethg++;
        Rks[6].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[7].ethg++;
        Rks[7].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[6].ethg++;
        Rks[7].iogate++ <--> Eth1G{per=0.008;}  <--> AS1[7].ethg++;
}
```

Sample C++ codes for Fat-tree DCN topology implementation in CloudNetSim++

The Fat-tree modelling in CloudNetSim++ can be illustrated briefly as followed,

```
module Racks_Fattree64
{
    parameters:
        int N @prompt("Number of servers per Rack");//input N=8
    gates:
        inout iogate[];
    submodules:
        computingServer[N]: myComputingNode
        edgeRouter: RouterGreenCloud
    connections:
        for i=0..N-1 {edgeRouter.ethg++ <--> Eth1G {per = 0.008;} <-->
computingServer[i].ethg++;}
```

```
        edgeRouter.ethg++ <--> iogate++;
        edgeRouter.ethg++ <--> iogate++;
}
module Datacenter_Fattree64
{
    parameters:
        int aggregateswitch = default(8);
          int corerouter = default(4);
          int accessswitch = default(8);
    gates:
        inout iogate[];
    submodules:
        core[corerouter]: RouterGreenCloud
        agg[aggregateswitch]: RouterGreenCloud
        Rack[accessswitch]: Racks_Fattree64
    connections allowunconnected:
        for j=0..1 {
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[0].ethg++;
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[2].ethg++;
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[4].ethg++;
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[6].ethg++;
        }
        for j=2..3 {
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[1].ethg++;
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[3].ethg++;
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[5].ethg++;
            core[j].ethg++ <--> Eth10G {per = 0.008;}<--> agg[7].ethg++;
        }
        for j=0..1, for i=0..1 {agg[j].ethg++ <--> Eth10G {per = 0.008;}<-->
Rack[i].iogate++;}
        for j=2..3, for i=2..3 {agg[j].ethg++ <--> Eth10G {per = 0.008;}<-->
Rack[i].iogate++;}
        for j=4..5, for i=4..5 {agg[j].ethg++ <--> Eth10G {per = 0.008;}<-->
Rack[i].iogate++;}
        for j=6..7, for i=6..7 {agg[j].ethg++ <--> Eth10G {per = 0.008;}<-->
Rack[i].iogate++;}
        core[0].ethg++ <--> Eth10G <--> iogate++;
}
```

Sample C++ codes for BCube-2 layer DCN topology implementation in CloudNetSim++

module MeshCDatacenter_Bcube

```
{
    parameters:
        int x @prompt("Number of L0 switch");
        int Level1switch = default(x);
        int Level0switch = default(x);
        int n @prompt("Number of server for each of L0");
        int Server = default(n*x);
    gates:
        inout iogate[];
    submodules:
        L1[Level1switch]: RouterGreenCloud
        L0[Level0switch]: RouterGreenCloud
        computingServer[Server]: myComputingNode
    connections:
        for j=0..x-1, for i=0..n-1 {L0[j].ethg++ <--> Eth10G <--> computingServer[(j*n)+i].ethg++;}
        for j=0..x-1, for i=0..n-1 {L1[j].ethg++ <--> Eth10G <--> computingServer[(i*n)+j].ethg++;}
}
```

Sample C++ codes for BCube-3 layer DCN topology implementation in CloudNetSim++

```
module MeshCDatacenter_Bcube2
{
    parameters:
        int Level2switch = default(16);
        int Level1switch = default(16);
        int Level0switch = default(16);
        int Server = default(64);
    gates:
        inout iogate[];
    submodules:
        L2[Level2switch]: RouterGreenCloud
        L1[Level1switch]: RouterGreenCloud
        L0[Level0switch]: RouterGreenCloud
        computingServer[Server]: myComputingNode
    connections:
        for j=0..15, for i=0..3 {L0[j].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[(j*4)+i].ethg++;}
        for j=0..15, for i=0..3 {L2[j].ethg++ <--> Eth10G {per=0.008;} <--> computingServer[(i*16)+j].ethg++;}
            L1[0].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[0].ethg++;
```

L1[0].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[4].ethg++;
L1[0].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[8].ethg++;
L1[0].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[12].ethg++;
L1[1].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[1].ethg++;
L1[1].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[5].ethg++;
L1[1].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[9].ethg++;
L1[1].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[13].ethg++;
L1[2].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[2].ethg++;
L1[2].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[6].ethg++;
L1[2].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[10].ethg++;
L1[2].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[14].ethg++;
L1[3].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[3].ethg++;
L1[3].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[7].ethg++;
L1[3].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[11].ethg++;
L1[3].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[15].ethg++;
L1[4].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[16].ethg++;
L1[4].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[20].ethg++;
L1[4].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[24].ethg++;
L1[4].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[28].ethg++;
L1[5].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[17].ethg++;
L1[5].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[21].ethg++;
L1[5].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[25].ethg++;
L1[5].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[29].ethg++;
L1[6].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[18].ethg++;
L1[6].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[22].ethg++;
L1[6].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[26].ethg++;
L1[6].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[30].ethg++;
L1[7].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[19].ethg++;
L1[7].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[23].ethg++;
L1[7].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[27].ethg++;
L1[7].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[31].ethg++;
L1[8].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[32].ethg++;
L1[8].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[36].ethg++;
L1[8].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[40].ethg++;
L1[8].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[44].ethg++;
L1[9].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[33].ethg++;
L1[9].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[37].ethg++;
L1[9].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[41].ethg++;
L1[9].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[45].ethg++;
L1[10].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[34].ethg++;
L1[10].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[38].ethg++;
L1[10].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[42].ethg++;
L1[10].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[46].ethg++;
L1[11].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[35].ethg++;

```
                L1[11].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[39].ethg++;
                L1[11].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[43].ethg++;
                L1[11].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[47].ethg++;
                L1[12].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[48].ethg++;
                L1[12].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[52].ethg++;
                L1[12].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[56].ethg++;
                L1[12].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[60].ethg++;
                L1[13].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[49].ethg++;
                L1[13].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[53].ethg++;
                L1[13].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[57].ethg++;
                L1[13].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[61].ethg++;
                 L1[14].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[50].ethg++;
                L1[14].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[54].ethg++;
                L1[14].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[58].ethg++;
                L1[14].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[62].ethg++;
                L1[15].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[51].ethg++;
                L1[15].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[55].ethg++;
                L1[15].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[59].ethg++;
                L1[15].ethg++ <--> Eth10G {per=0.008;}<--> computingServer[63].ethg++;
}
```

Sample C++ codes for realizing Energy Consumption Management

```
void UDPComputeNode::eUpdate() /* complete impl*/
{
      /* Get time spent since last update */
        double etime = (simTime().dbl() - eLastUpdateTime_)/3600;      /* time in
hours */
        eConsumed_ += etime * eCurrentConsumption_;
        eLastUpdateTime_ = simTime().dbl();
}
void UDPComputeNode::setCurrentConsumption()    /* complete impl*/
{
      /* Compute idle server consumption */
         double eIdleConsumption = eNominalrate_*2/3;
         /* if DPM is enabled no energy is consumed with zero load */
         if ((getCurrentLoad() == 0) && (eDPM_enabled_)) {
            eCurrentConsumption_ = 0;
              return;
         }
         /* if DVFS is enabled energy consumed is scaled with the frequency */
         if (eDVFS_enabled_) {
```

```
            double f = getCurrentLoad();        /* frequency component */
            eCurrentConsumption_ = eIdleConsumption + eNominalrate_ * f*f*f / 3;
            return;
          }
          /* Compute load dependant energy consumption component */
          double eLoadComponent = (eNominalrate_ - eIdleConsumption) *
getCurrentLoad();
          eCurrentConsumption_ = eIdleConsumption + eLoadComponent;
}
double UDPComputeNode::getMostUrgentTaskRate()
{
       std::vector<cloudTask*>::iterator iter;
         /* Compute highest MIPS/deadline ratio */
         double maxrate = 0.0;
         /* remove completed tasks from the execution list */
         for (iter = tasks_list_.begin(); iter != tasks_list_.end(); iter++)
         {
            /* task should be completed and remove it from the list */
            double rate = (double)(*iter)->getMIPS()/(double)(*iter)->getDeadline();
            if (rate > maxrate) maxrate = rate;
         }
         return maxrate;
}
double UDPComputeNode::getCurrentLoad()
{
       ev<<"Value of current_mpis"<<current_mips_<<endl;
       ev<<"Value of nominal_mpis"<<nominal_mips_<<endl;
       currentLoad_ = (double)current_mips_/(double)nominal_mips_;
       return currentLoad_;
}
```

Appendix B: Gephi Network Analysis Tool environment

The distributed DCNs can be illustrated as follows in Gephi, there are four geographical fashioned DCNs interconnected by five main routers which are arranged in the middle of the diagram.

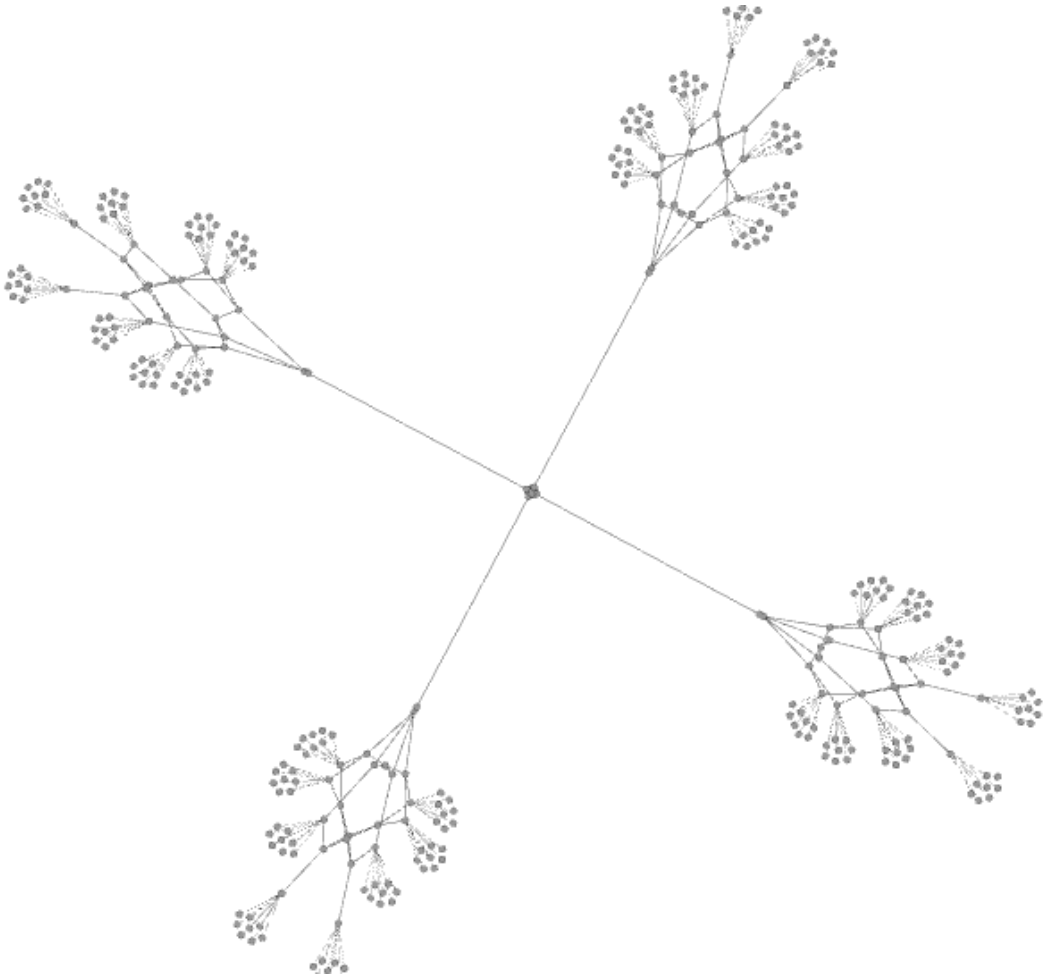


Figure 0.1 – Appendix: Gephi-diagram of 4 distributed DCN

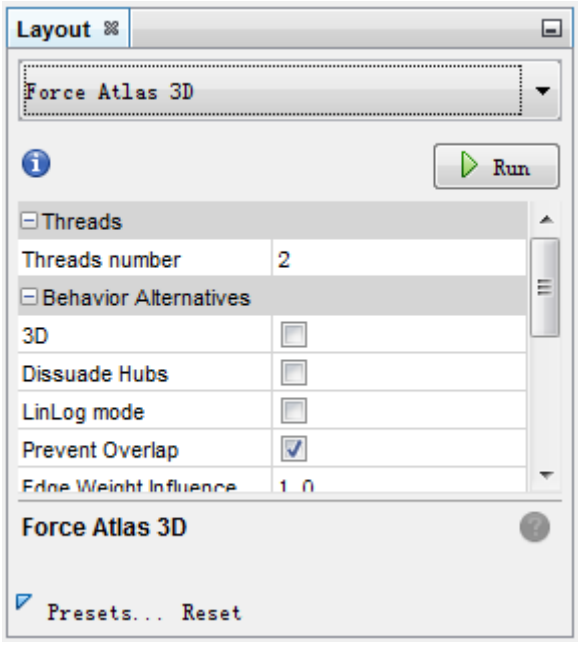


Figure 0.2 – Appendix: Gephi-Layout panel

The layout panel supports nodes and links arrangement with a 3D render engine. It offers several choices that aim at various interconnection ways.

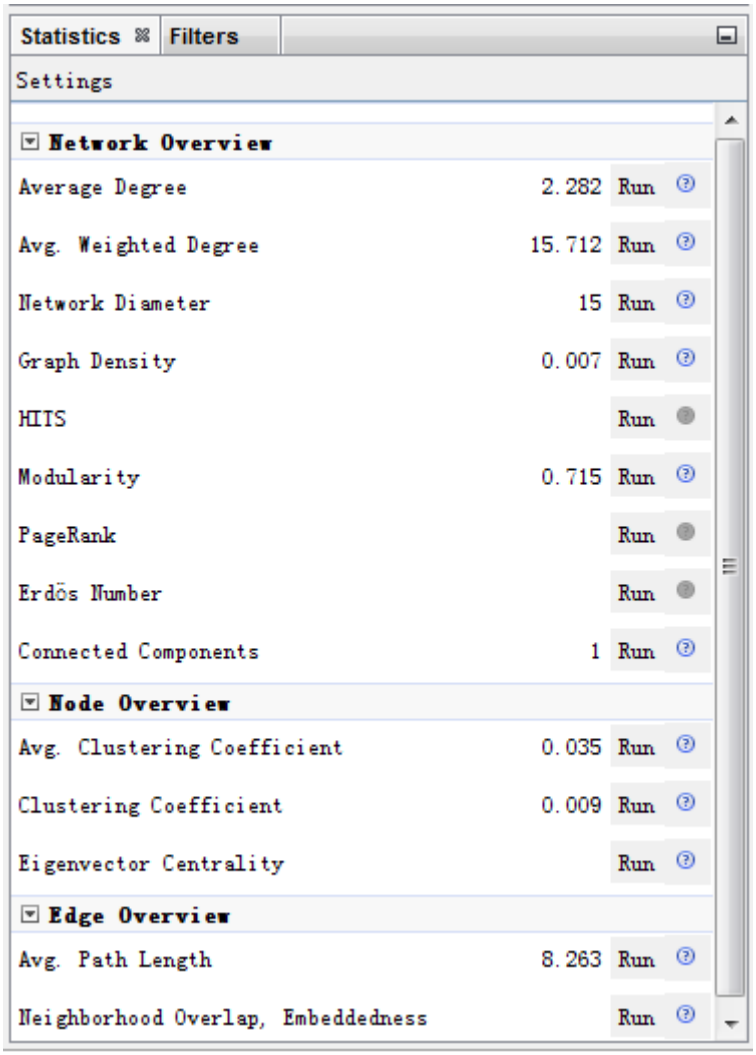
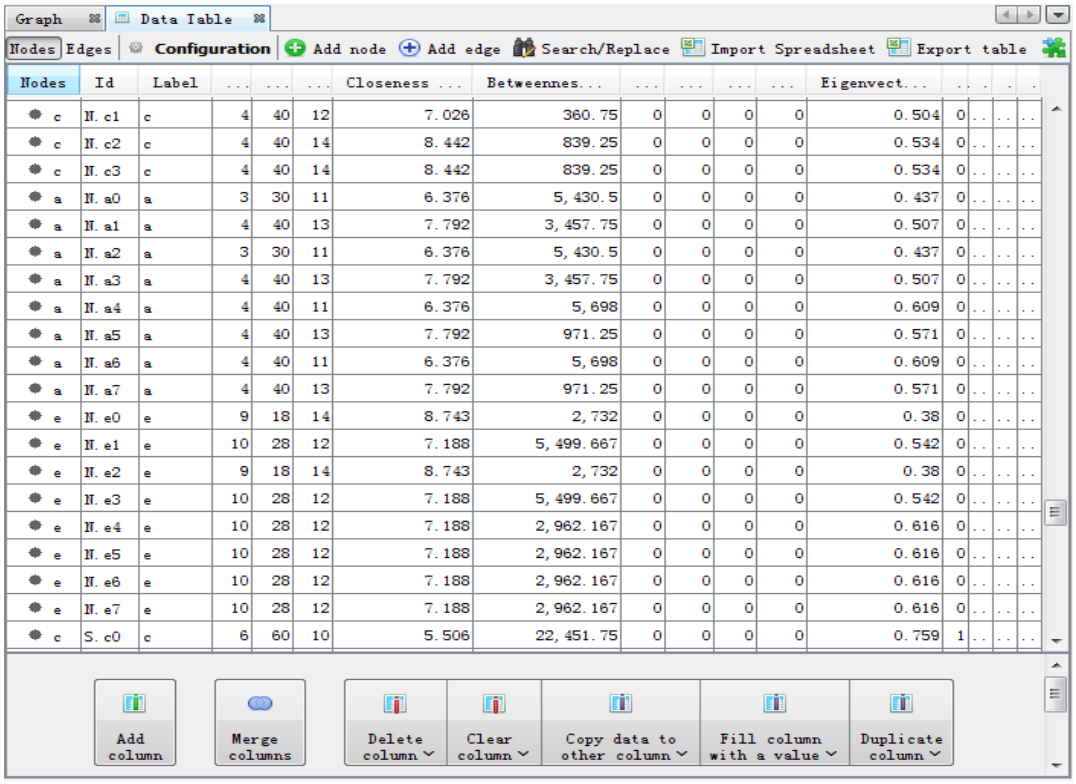


Figure 0.3 - Appendix: Gephi-Statistics panel

The statistics panel supports the measurements of a series of network robustness metrics, the betweenness centrality, closeness centrality are placed inside the Avg. Path Length.



Nodes	Id	Label	...	...	...	Closeness ...	Betweennes...	...	...	...	...	Eigenvect...	...	...	...	...
c	N. c1	c	4	40	12	7. 026	360. 75	0	0	0	0	0. 504	0	..	..	..
c	N. c2	c	4	40	14	8. 442	839. 25	0	0	0	0	0. 534	0	..	..	..
c	N. c3	c	4	40	14	8. 442	839. 25	0	0	0	0	0. 534	0	..	..	..
a	N. a0	a	3	30	11	6. 376	5, 430. 5	0	0	0	0	0. 437	0	..	..	..
a	N. a1	a	4	40	13	7. 792	3, 457. 75	0	0	0	0	0. 507	0	..	..	..
a	N. a2	a	3	30	11	6. 376	5, 430. 5	0	0	0	0	0. 437	0	..	..	..
a	N. a3	a	4	40	13	7. 792	3, 457. 75	0	0	0	0	0. 507	0	..	..	..
a	N. a4	a	4	40	11	6. 376	5, 698	0	0	0	0	0. 609	0	..	..	..
a	N. a5	a	4	40	13	7. 792	971. 25	0	0	0	0	0. 571	0	..	..	..
a	N. a6	a	4	40	11	6. 376	5, 698	0	0	0	0	0. 609	0	..	..	..
a	N. a7	a	4	40	13	7. 792	971. 25	0	0	0	0	0. 571	0	..	..	..
e	N. e0	e	9	18	14	8. 743	2, 732	0	0	0	0	0. 38	0	..	..	..
e	N. e1	e	10	28	12	7. 188	5, 499. 667	0	0	0	0	0. 542	0	..	..	..
e	N. e2	e	9	18	14	8. 743	2, 732	0	0	0	0	0. 38	0	..	..	..
e	N. e3	e	10	28	12	7. 188	5, 499. 667	0	0	0	0	0. 542	0	..	..	..
e	N. e4	e	10	28	12	7. 188	2, 962. 167	0	0	0	0	0. 616	0	..	..	..
e	N. e5	e	10	28	12	7. 188	2, 962. 167	0	0	0	0	0. 616	0	..	..	..
e	N. e6	e	10	28	12	7. 188	2, 962. 167	0	0	0	0	0. 616	0	..	..	..
e	N. e7	e	10	28	12	7. 188	2, 962. 167	0	0	0	0	0. 616	0	..	..	..
c	S. c0	c	6	60	10	5. 506	22, 451. 75	0	0	0	0	0. 759	1	..	..	..

Figure 0.4 - Appendix: Gephi-Data Table

The data is imported by csv formatted file in this case. Nodes and edges are inputted in separate files. In the Gephi data table, the calculated statistics are shown by nodes.

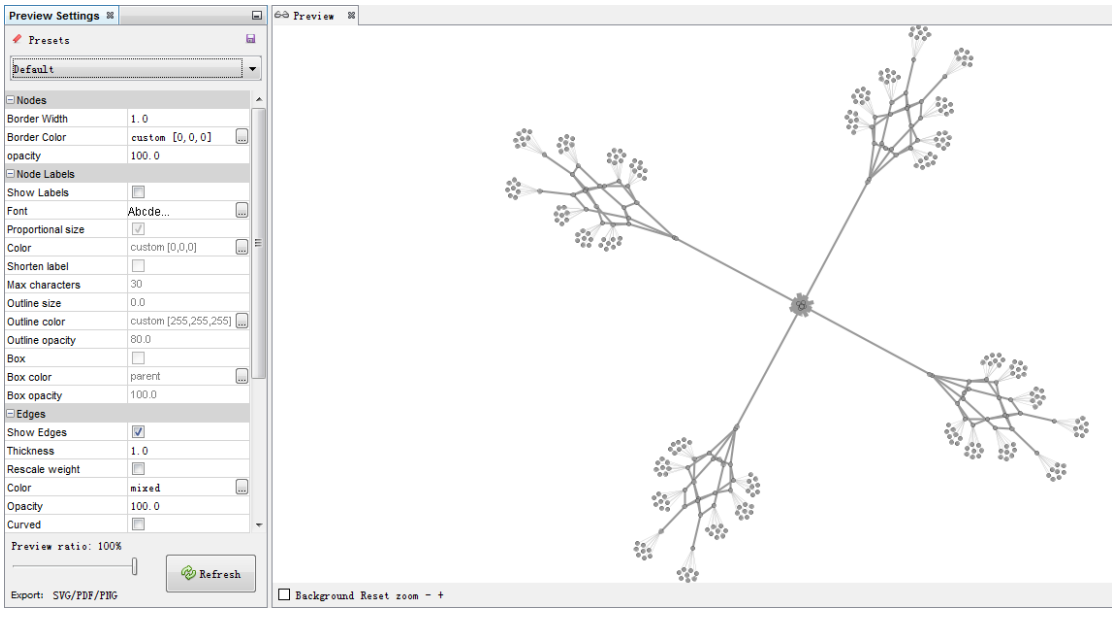


Figure 0.5 – Appendix: Gephi-Preview panel

In the “Preview” section, parameters can be set up by preferences include label, proportional size, color, font type, etc...