UNIVERSITY OF BURGUNDY
U.F.R. SCIENCES ET TECHNIQUE
ECOLE DOCTORAL SPIM

THESIS

Presented by:

Emna Baccour

Submitted in fulfilment of the requirements
for the degree of:

DOCTOR OF THE UNIVERSITY OF BURGUNDY

NETWORK ARCHITECTURES AND ENERGY EFFICIENCY FOR HIGH PERFORMANCE DATA CENTERS

ARCHITECTURES RÉSEAUX ET OPTIMISATION D'ÉNERGIE POUR LES CENTRES DE DONNÉES MASSIVES

Defended on 30-06-2017

Jury :

Mme. Nadia BOUKHATEM, Professeur, Telecom ParisTech, Rapporteur.

M. Hamameche KHEDDOUCI, Professeur, Université Claude Bernard Lyon 1, Rapporteur.

M. Bernard TOURANCHEAU, Professeur, Université Grenoble Alpes, Examinateur.

M. Sebti FOUFOU, Professeur, Université de Bourgogne, Directeur de thèse.

M. Ridha HAMILA, Professeur, Université de Qatar, Co-encadrant.

"To my beloved grandfather, although not with us, your belief in me has made this success possible. you are gone, but you left fingerprints of grace in my life."

Abstract

The increasing trend to migrate applications, computation and storage into more robust systems leads to the emergence of mega data centers hosting tens of thousands of servers. As a result, designing a data center network that interconnects this massive number of servers, and providing efficient and fault-tolerant routing service are becoming an urgent need and a challenge that will be addressed in this thesis. Since this is a hot research topic, many solutions are proposed like adapting new interconnection technologies and new algorithms for data centers. However, many of these solutions generally suffer from performance problems, or can be quite costly. In addition, devoted efforts have not focused on quality of service and power efficiency on data center networks. So, in order to provide a novel solution that challenges the drawbacks of other researches and involves their advantages, we propose to develop new data center interconnection networks that aim to build a scalable, cost-effective, high performant and QoS-capable networking infrastructure. In addition, we suggest to implement power aware algorithms to make the network energy effective. Hence, we will particularly investigate the following issues: 1) Fixing architectural and topological properties of the new proposed data centers and evaluating their performances and capacities of providing robust systems under a faulty environment. 2) Proposing routing, load-balancing, fault-tolerance and power efficient algorithms to apply on our architectures and examining their complexity and how they satisfy the system requirements. 3) Integrating quality of service. 4) Comparing our proposed data centers and algorithms to existing solutions under a realistic environment.
In this thesis, we investigate a quite challenging topic where we intend, first, to study the existing models, propose improvements and suggest new methodologies and algorithms.

Keywords: data center, quality of service, routing, scalability, power efficiency.

Résumé

L'évolution des services en ligne et l'avènement du big data ont favorisé l'introduction de l'internet dans tous les aspects de notre vie : la communication et l'échange des informations (exemple, Gmail et Facebook), la recherche sur le web (exemple, Google), l'achat sur internet (exemple, Amazon) et le streaming vidéo (exemple, YouTube). Tous ces services sont hébergés sur des sites physiques appelés centres de données ou data centers qui sont responsables de stocker, gérer et fournir un accès rapide à toutes les données. Tous les équipements constituants le système d'information d'une entreprise (ordinateurs centraux, serveurs, baies de stockage, équipements réseaux et de télécommunications, etc) peuvent être regroupés dans ces centres de données. Cette évolution informatique et technologique a entrainé une croissance exponentielle des centres de données. Cela pose des problèmes de coût d'installation des équipements, d'énergie, d'émission de chaleur et de performance des services offerts aux clients. Ainsi, l'évolutivité, la performance, le coût, la fiabilité, la consommation d'énergie et la maintenance sont devenus des défis importants pour ces centres de données. Motivée par ces défis, la communauté de recherche a commencé à explorer de nouveaux mécanismes et algorithmes de routage et des nouvelles architectures pour améliorer la qualité de service du centre de données. Dans ce projet de thèse, nous avons développé de nouveaux algorithmes et architectures qui combinent les avantages des solutions proposées, tout en évitant leurs limitations. Les points abordés durant ce projet sont: 1) Proposer de nouvelles topologies, etudier leurs proprietés, leurs performances, ainsi que leurs coûts de construction. 2) Conception des algorithmes de routage et des modèles pour réduire la consommation d'énergie en prenant en considération la complexité, et la tolérance aux pannes. 3) Conception des protocoles et des systèmes de gestion de file d'attente pour fournir une bonne qualité de service. 4) Évaluation des nouveaux systèmes en les comparants à d'autres architectures et modèles dans des environnements réalistes.

Mot clés: centres de données, algorithmes de routage, qualité de service, consommation d'énergie.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Sebti Foufou for his kindness, help, support and guidance. You have offered me the most valuable advises through all these three years. Your comprehensive research methodologies and dedicated research attitudes have benefited me a lot.

I would like also to thank Prof. Ridha Hamila for his important contributions, comments and ideas. At many stages of this thesis I benefited from his advice. I express my deep and sincere gratitude.

Prof. Nadia Boukhatem and Prof. Hamamache Kheddouci have made me honored by reviewing my thesis. I would like also to thank Prof. Bernard Tourancheau for being a member of my Jury.

All my thanks also go to my teammate and lovely friend Zina Chkirbene. Being a member of a research team with her for three years has been a wonderful experience.
Special thanks go to my friends Abir and Raghda, my cousin Zeineb and all of the others I do not mention here who definitely deserve my sincere acknowledgments.

I wish to give my heartfelt thanks to my very special person, my husband Ahmed, for his love, understanding, unlimited patience, and constant source of support and encouragement during the challenges of this thesis. You were always around at times I thought that it is impossible to continue. I am very grateful to have you in my life.

I would like also to thank my parents, grandmother and my brother for giving me the love, strength and patience to work through all these years so that today I can stand proudly with my head held high. I will never be able to pay back this love and sacrifice.

Contents

List of Figures

List of Tables

Abbreviations

ABT	**A**ggregate **B**ottlneck **T**hroughput
APL	**A**verage **P**ath **L**ength
BI	**B**locking **I**sland
CPU	**C**entral **P**rocessing **U**nit
DC	**D**ata **C**enter
DCN	**D**ata **C**enter **N**etwork
DFS	**D**epth **F**irst **S**earch
DVFS	**D**ynamic **V**oltage and **F**requency **S**caling
HLSDB	**H**istorical **L**ink and **S**tate **D**ata **B**ase
IP	**I**nternet **P**rotocol
IT	**I**nformation **T**echnology
LF	**L**east **F**low
LL	**L**east **L**ink
LOS	**L**ine **O**f **S**ight
Mac	**M**edia **a**ccess **c**ontrol
NIC	**N**etwork **I**nterface **C**ard
NSA	**N**etwork **S**tate **A**dvertisement
NSDB	**N**etwork **S**tate **D**ata **B**ase
OE	**O**pt **E**dge
OSPF	**O**pen **S**hortest **P**ath **F**irst protocol
PCA	**P**rincipal **C**omponent **A**nalysis
QoS	**Q**uality **o**f **S**ervice
R	**R**andom
SDN	**S**oftware **D**efined **N**etwork
ToR	**T**op **o**f **R**ack

VM	**V**irtual **M**achine
VOVO	**V**ary **O**n **V**ary **O**ff

Chapter 1

Introduction

In the last few years, data centers become the backbone of the world business, economy, communication, and consumer services. All online activities, including web searching (e.g., Google), social networking (e.g., Facebook), video streaming (e.g., Youtube), gaming (e.g., Steam), shopping (e.g., Amazon) are hosted in large data centers. However, few decades ago, business relied only on paper and pencil documentation methods. Data centers improved largely the creation, the usability, the maintenance and the storage of the data. In this way, with a browser or a software installed on any device, we can use data center services freely.

1.1 Data Centers evolution

The concept of data centers (DCs) was proposed in 1990. However, the features and requirements of the data centers appeared in the early 1960s with the first computer operations deployed in a computing center of some university laboratories. During the 1980s, people started to deploy computers everywhere but with no care about quality of service and operating requirements[4]. Then, the computing operations became more and more complex and the organizations became aware that the information technology (IT) needs to be controlled. Hence, since network equipment were inexpensive, specific rooms inside the companies were built and an hierarchical design was proposed. In this time, the term "data center" began to be used and studies were conducted in this field. The boom of data centers came between 1997 and 2000 where companies needed fast internet connection and high performance to deploy heavy applications. Therefore, they migrated to private networks. New technologies and practices were designed also to handle the scale and the requirements of the network. Then, as computation moved into the cloud, Google CEO proposed that the data services and

architectures should be on the cloud. After that, Amazon installed its web services on the cloud and Wikipedia introduced the virtualization on its networks as a service over cloud. Nowadays, the term "cloud" is integrated into the term "data center" and data centers begin to play more and more important role in web searching, social networking, large scale computations, online gaming and so on.

1.2 Data Center design challenges

To support the growing needs of cloud computing need, such data centers should host hundreds of thousands of servers, processing different types of services such as MapReduce[5], GFS[6], BigTable[7]. This huge number of servers is growing every day[8]. For example, Microsoft is doubling the number of servers in its data centers every 14 months[9], which is faster than Moore's Law[10]. This is resulting in enormous challenges to design a robust interconnected network with a cost-effective deployment and maintenance. In addition, with data availability and security at stake, the robustness of data center network becomes more critical than ever. Generally, the design goals of data center networks are: high scalability, good fault tolerance, low latency, high network capacity, load balancing, low cost, energy consumption and virtualization if necessary.

High scalability: A scalable system is a system that meets three requirements: first, the network must be able to connect millions of servers at a reasonable cost, simple installation and small number of wires. Second, the scalability must be gradual. It means the topology must be able to offer different sizes of the network. Third, the routing algorithm must be itself scalable and efficient to be processed in a large network.

Fault tolerance: With a growing number of equipment and cables in data center network, failures can become common rather than exception. A robust data center is the one that continues working without being affected by failures due to its fault-tolerant routing algorithm and its design that offers multiple connections between nodes.

Low latency and short path length: To offer faster services, communication between servers must be as rapid as possible. So, a short path length between any two nodes must be provided by the network design. The low latency must remain valid even for the large scale data centers.

High capacity: large data centers composed of millions of servers need high bandwidth to offer a sufficient runtime performance. As an example, The mapReduce is one of the applications that requires a large amount of bandwidth. Indeed, in the reduce operation, the reduce agents communicate with many servers in the network. So, an all-to-all communication is adopted by this application and requests consequently a high network capacity.

Load balancing: load balancing techniques are conceived to improve the distribution of workloads across multiple computing resources. It ensures better link utilization and higher throughput and capacity.
Cost: A convenient data center should interconnect a large number of servers using the minimum number of network equipment (switches, cables, ...). Hence, the simpler the architecture is, the less cost the network has.
Power consumption: a green data center is a network that consumes energy proportionally to the traffic workload and meets in the same time reliability and high performance requirements.
Virtualization: a well virtualized network is the one where virtual machines (VMs) can migrate to any physical machine without changing the IP address. This migration should be also well supported which means the performance should not be affected.

1.3 Contributions

Many research efforts focused on finding both efficient network interconnections that satisfy QoS requirements as well as satisfying the requirements of data center networks. The followings are some examples of well known data center designs: FatTree[11], DCell[9], BCube[12], ElasticTree[13], and wireless topologies[14]. However, gathering all data center requirements in one topology was not addressed by these solutions. Existing solutions either scale too fast (but suffer from a large latency and a long path length), offer a high performance in term of latency (but fail to construct a scalable topology), inherit poor availability, or can be too complex/expensive to be constructed. In this thesis, we address these aforementioned issues by conducting series of analyses and developing novel algorithms/architectures, where the ultimate goal is to build a scalable, high capacity, energy efficient and cost-effective data center networking infrastructure. The contributions that we have made are summarized as follows:

1.3.1 Enhance the latency: Wireless technology (wFlatnet)

First, we tackled two challenges which are latency and average path length taking into consideration the load balancing and fault tolerance properties. An intuitive idea would be to add some switches and links to the network which may reduce the path length at a price of high complexity of wiring and installation not to mention the cost of cabling that represents 7-8 % of the overall infrastructure cost[14]. So, our contribution is to add wireless links to existing networks as a solution to reduce the transmission path. This new designed topology, called wFlatnet, introduces shortcuts between

nodes through the wireless connections without burdening the network with wiring and maintenance complexity.

1.3.2 Enhance the scalability: Paramatrizable topology (PTNet)

Second, we handled the problem of scalability. Thus, we proposed a novel server-centric network topology, called PTNet, characterized by its parameterizable architecture that allows constructing a scalable or a small network according to the applications need. In addition, PTNet makes a trade-off between scalability, low latency and power consumption. To meet such a goal, PTNet enjoys a new network interconnection that connects servers directly by using their capacity of routing packets. This approach reduces the number of switches and consequently reduces path length, packet delay and the cost of the extra-machines. PTNet rich physical connection allows to have different routes between nodes to improve the performance and fault-tolerance of the system. It is also based on clusters to build its topology. The size of each cluster can be adjusted by a parameter s to make PTNet a parameterisable architecture that constructs different network sizes.

1.3.3 Greening the network: Power aware approaches

Finally, we investigated the problem of power consumption in data centers. In fact, it is noted that a typical data center operates only at 5% to 25% of its maximum capacity depending on the period where the utilisation of the network fluctuates according to the incoming loads[13][15]. If the load is low, the servers are kept idle and they may still consume up to 70% of their peak power which causes a great waste of energy[16]. This situation is worsened by the traditional routing algorithms that do not take into consideration the non-proportionality between the traffic load and the energy consumption. Based on these observations, we can deduce that when the network is under-utilized, the traffic can be satisfied by only a subset of devices and the idle ones can be powered off to save the energy. In this thesis, we focused on designing power aware routing algorithms that switches on only ports of servers contributing to the traffic load and the critical nodes maintaining the system non-partitioned. Then unneeded devices are deactivated. Two routing level approaches were proposed to handle the power inefficiency in the network. But, since these two solutions are traffic load dependent which is a random factor, we proposed, also, a queuing model approach to make greening the network independent from the traffic matrix. The idea is to consolidate the arriving traffic into few ports of the networking device and switch off the rest of interfaces. In this way, we do not need to know the traffic matrix and to define the idle devices.

1.4 Thesis organization

This thesis is organized as follows:

- Chapter 2 includes a literature review on recent data center networks with their classification.
- Chapter 3 characterizes and describes two Data Center Interconnection networks (DCNs) termed wFlatnet and PTNet respectively. This chapter covers, also, the simulation of the new data centers followed by the performance results.
- Chapter 4 presents two routing aware algorithms that aim at making the power consumption proportional to the traffic load.
- Chapter 5 introduces a theoretical analysis of the queuing model proposed to make the network green and independent from the traffic load. An optimization approach to ensure the performance of the data center is also described.
- Chapter 6 presents the general conclusion and gives some ideas for future works.

Chapter 2

State of the art

2.1 Introduction

Recent researches have investigated various scientific challenges related to data center networks and addressed many of networking issues. A lot of innovative ideas are proposed such as creating novel data center designs, enhancing the performances of existing topologies by adding wireless links or virtual machines, proposing algorithms to make the network power efficient, etc. Based on these efforts, we have conducted, in this chapter, a deep study about existing data centers and investigated related works to describe the faced challenges, proposed solutions and critics that helped us to propose few contributions.

2.2 Examples of existing data centers

In this section, we briefly present data centers of two big companies: Google and Microsoft.

2.2.1 Google DCs

Google data centers are the softwares and large hardware resources used by Google to offer their services to the clients. The first prototype of Google data center is called Backhub (see Figure 2.1(a)) and was placed in the dorm of one of the founders of Google, Larry Page. This prototype was simple. However, it met all the requirements of Google searching. Now, Google owns 36 data centers located in all over the world:

19 in the USA, 12 in Europe, 3 in Asia, 1 in Russia and 1 in South America. Additionally, more than 10 data centers will be built before 2019[17]. The first scalable data center was constructed in Portland Dalles (see Figure 2.1(b)), costed 600 $ millions, occupied an area of 94 000 square feet and powered by Dalles dam. Then, another 600 $ was spent, in 2013, to build a second data center in Dalles occupying 164 000 square feet. Searching, emailing and maps are offered by Douglas data center which uses the recycled water system to provide cooling needs. Finland data center uses also environmental resources (sea water) to control the temperature of its buildings. As energy cost is the top priority of Google, Belgium data center, which costed 250 € millions, utilizes the gray water (clean water waste) to power its cooling systems. By using the renewable energy, Google electrical power of its global operations ranges between 500 and 681 megawatts [18][4].

(a) (b)

FIGURE 2.1: Google data centers.

Google uses commodity servers with a customized version of Linux. Their CPUs offer the best performance per dollar and not the absolute performance. The servers are open from the top so that a rack (see section 2.3.4) can contain more devices. Each server has an additional 12-volt battery to reduce the cost and the power consumption. Concerning the software, most of them are developed in-house.
Details of Google worldwide networks topology are not publicly available but Google publications make references to the Clos architecture (see section 2.5.1.1) and recently the software defined networks (SDN).

2.2.2 Microsoft DCs

Microsoft [19] owns many data centers worldwide: USA, Europe and Asia. Washington Quincy hosts 2 data centers: the first one covers an area of 75 acres (see Figure 2.2(a)) and the second one, which is a modular data center (the basic component of a DC is

a shipping container), covers 93 000 square feet. Microsoft owns also a data center in Texas San Antonio occupying half a million square feet and costing 550 $ millions. This data center uses the recycled water to control the temperature. The biggest data center of Microsoft, which is also one of the biggest DCs in the world, is located in Chicago, costs 500 $ millions and covers more than 700 000 square feet (see Figure 2.2(b)) . The biggest oversea data center is placed in Dublin, Ireland covers 303 000 square feet and uses the natural wind for cooling purposes.
Microsoft DCs connect more than 1 million servers and support more than 200 online services such as Azure, skype, oneDrive, Bing, Office and Xbox live. More than 15000 $ millions are invested to build 1600 unique networks and 1.4 million miles of fibers.
Microsoft network architecture is a Clos-based design. The company adopts also the virtualization in several components of the architecture in order to enhance the scalability, the flexibility and the CPU utilization.
Nowadays, approximately 44% of the electricity consumed by Microsoft data centers are produced from wind, solar and hydraulic energy. The target goal is to reach 50% of utilization of renewable energy by 2018. In addition, Microsoft achieves 100% of carbon neutrality since 2012.

(a) (b)

FIGURE 2.2: Microsoft data centers.

2.3 Hardware of data center networks

A data center consists of several physical components including: switches, servers, storage, racks, cables, cooling machines and power production as presented in Figure 2.3.

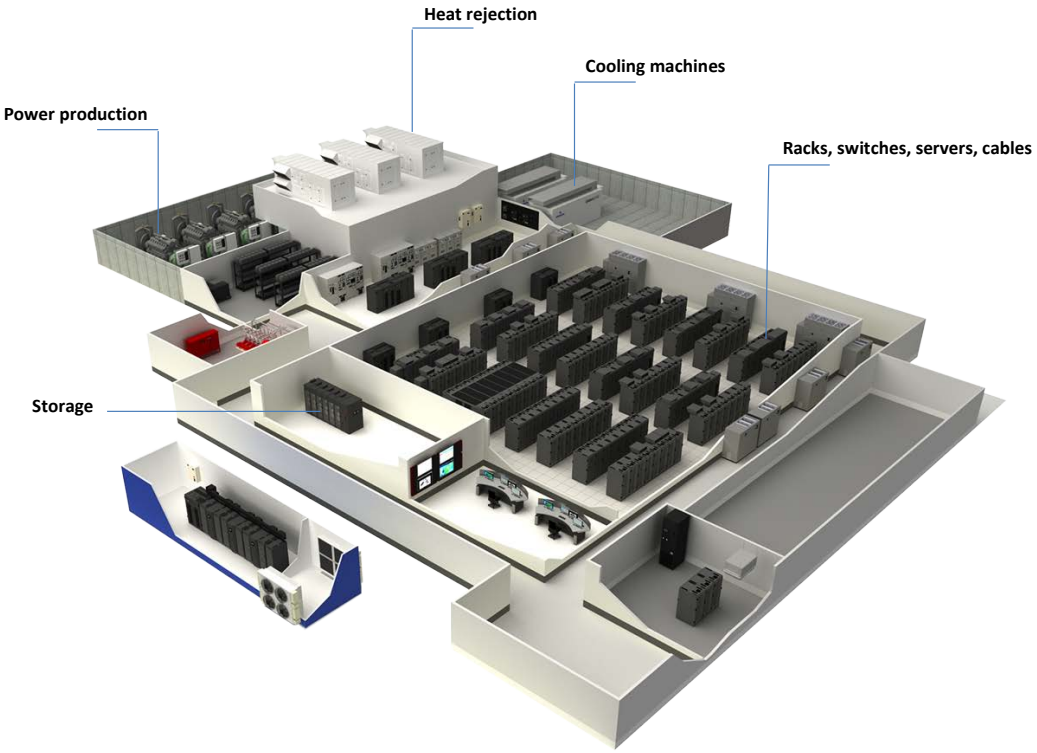


FIGURE 2.3: Data center.

2.3.1 Switch

The switch is the principal component of many data center architectures such as the three-layered topologies and the virtualized networks. In fact, all the switches have a basic functionality which is providing a media access control (Mac) addresses to the ports. These ports are responsible for forwarding packets to the intended destinations, prevent network loops and segment the traffic. However, different types of switches have specific characteristics. For the tree-layered architectures, we distinguish 3 types: core, distribution and access switches[20]. Core switches are characterized by their speed, high throughput and high performance. Distribution switches connect core and access switches and aim to choose the optimal path to relay packets and have better quality of service and high memory. Access switches, which are directly connected to the end-users, are better from security perspective. However, since data center architectures are changing (eg. the growth of virtualized networks), new DC-class switches are continuously being designed. They are characterized by their higher availability, better fault tolerance, higher flexibility, compatibility with other configurations and easier control.

2.3.2 Server

Servers are the core of data center architectures that store, process, transmit and analyze the data. They can be categorized into three types: tower, rack and blade[21]. Tower servers are the first ones used in data centers. However, they can only satisfy the requirements of a small network. It means they are not appropriate for a high scale data center. Hence, rack servers are designed for modern DCs. They are highlighted by their space saving and their easy placement into a rack. In this way, a rack can contain multiple racks servers. However, they suffer from high cabling complexity. Blade servers are designed to overcome the drawbacks of the aforementioned servers. They are characterized by a low cost, high availability, reduced maintenance and simple installation. Even though data center servers are standardized, IT companies such as Facebook and Google customize their servers software.

2.3.3 Storage

Data center storage refers to the devices, tools, technologies and software used to implement, manage and control the storage resources such as hard disk drivers, backup utilities, external storage, etc. It also includes the policies, procedures and security methodologies of the storage. Small data centers adapt a centralized storage management. However, nowadays, the stored data is increasing rapidly (photos, voice, videos, etc). In this case, the centralized management is not suitable for the cloud data. Hence, a distributed storage management is adapted (eg. Amazon storage, Windows Azure storage). Another trend is the Software Defined Storage which is a form of storage virtualization that separates the storage hardware from the software that manages it[21][22].

2.3.4 Rack

A rack in a data center is designed to house network equipment (rack servers, blade servers, switches and storage devices). Using the rack design, the management of the DC equipment is easier and the use of the room space can be more efficient. There are two types of racks commonly used in a data center which are the open racks and the cabinet racks. The open racks are easier to manage and install. However, the cabinets are more stable and secure. The dimensions of a rack are generally: height is between 42 U and 48 U (1U=44.45mm), width is between 600 and 800 mm and depth is between 1100 and 1200 mm [21]. Racks are placed in rows forming corridors (called aisles) between them to allow technicians to access to the front and near of each cabinet.

2.3.5 Cable

Nowadays, data centers house a huge number of networking devices including blade servers, efficient storage, virtualized devices that require all a physical cabling with a high performance, low cost, easy installation, reliability, high scalability and flexibility. In data centers, two methods of cabling are used: the backbone cabling and horizontal cabling. The backbone cables generally connect the equipment rooms and telecommunication rooms. However, the horizontal cables connect individual outlets to telecommunication rooms. The Ethernet standards used in DCs are: 10GBASE, 40GBASE and 100GBASE[23]. Cables represent 7-8 % of the overall data center investment[14].

2.3.6 cooling machines

The air conditioning is used to control the temperature and the humidity in data centers. The temperature in a data center room is generally raised because of electronic equipment[24]. By controlling the air temperature, the networking equipment are maintained within the specified humidity/temperature range. Modern data centers are trying to decrease the cost of cooling by adapting new techniques such as outside air cooling, sea water cooling and renewable energy to operate air conditioner.

2.3.7 Power production

The power production part of a data center building consists of power supplies, gaz generators and battery banks. To prevent power failures, all electrical system elements are duplicated[24].

2.4 Definitions

To present existing data centers and the studies conducted in this field, some parameters should be defined as they will be used in further sections.

2.4.1 Average path length

The average path length (APL) is the average number of hops traveled by packets to reach the destinations. This metric is important to evaluate the efficiency of the transmission and the communication delay. It can be computed as follows:

$$APL = \frac{\sum\limits_{i,j} l_{i,j}}{\sum\limits_{i,j} \rho_{i,j}} \tag{2.1}$$

where $l_{i,j}$ represents the path length between the two nodes i to j, with $\rho_{i,j} = 1$ if a routing path exists between i and j and $\rho_{i,j} = 0$ if not.

2.4.2 Diameter

The diameter represents the maximum shortest path between any two servers in the network. A smaller diameter helps to have a more effective routing algorithm and a lower latency.

2.4.3 Throughput

The throughput of the network depends on the packet delay, the data rate of the channel and the rate of successful messages. Having multiple possible routing paths between nodes leads to less traffic congestion and more available bandwidth which improves the throughput. The throughput can be obtained as follows:

$$T_{avg} = \frac{1}{n_p} \sum_{i=1}^{n_p} (\frac{\rho_i * \delta_i}{d_i}) \tag{2.2}$$

where T_{avg} is the average throughput of the network, $\rho_i \in [0, 1]$, with $\rho_i = 1$, represents the success of the reception of a packet i and $\rho_i = 0$ represents the failure of the reception of a packet i, δ_i is the size of the packet i, d_i is the delay of the transmission of the packet i and n_p is the total number of transmitted packets.

2.4.4 Average network latency

Many data centers applications have to meet various time constraints[25]. Therefore latency is an important metric to judge whether these constraints are met or not. Latency is defined as time taken by the packet to travel from a source to destination. It

consists of the transmission delay, the propagation delay, the queuing delay and the time to process data in servers and switches. However, the major contributor in the packet delay delivery is the queuing and processing of the data at each relay node. The average packet delay, called also the average network latency, is computed as follows:

$$D_{avg} = \frac{1}{n_p} \sum_{i=1}^{n_p} d_i \tag{2.3}$$

where D_{avg} is the average latency of the system, n_p is the total number of packets communicated in the system and d_i is the delay of the communication between two nodes when sending a packet i.

2.4.5 Aggregate bottleneck throughput

The aggregate bottleneck throughput (ABT) is used to evaluate the network capacity under all-to-all traffic pattern (all servers send data to all other servers). The flows that have the smallest throughput are called bottleneck flows. The aggregate bottleneck throughput is obtained by summing all bottleneck flows.

2.4.6 Bisection width

The bisection width represents the minimum number of links cut when the network is partitioned into two equal halves. It is used to estimate the worst failure case of the system. Hence, the larger the bisection width is, the better fault tolerance the network will have.

2.4.7 Oversubscription

The oversubscription is the worst case achievable bandwidth among the devices. An oversubscription equal to 1:1 means that all servers communicate with the full bandwidth and an oversubscription equal to 5:1 means that 20% of the bandwidth is available for the traffic communications.

2.4.8 Degree of the server

The degree is the number of connected ports per server. A standard server has 2 ports. However, some networks add ports to servers in order to make the network scalable.

2.4.9 Number of switches, wires and servers

We assume in this thesis that all switches, wires and servers are the same so we can have a fair comparison between topologies. The number of servers determines the scalability of the network. The DC that connects more servers with the same number of network equipment (switches with same number of ports, wires, etc.) is the most scalable network.

2.4.10 Disjointed node/edge paths

The number of disjointed node paths represents the total number of paths between any two servers, where these paths do not have any common intermediate nodes. This metric could be used to assess whether a given network is well designed as well as having required physical connections. The more number of disjointed paths a given network has, the more fault tolerant it becomes: if one (routing) path fails, the other paths can be used. The number of disjointed edge paths represents the total number of paths between any two servers, where these paths do not have any common intermediate edges.

2.4.11 Traffic matrix/pattern

The traffic matrix is the table of communicating servers (sources and destinations) at a time t.

2.5 Architectures of data center networks

Data centers are composed of interconnected physical components to support cloud services. The number of connected devices is increasing everyday and reaching even millions. Such a huge network should provide a good quality of service and needs to be reliable, scalable, flexible, robust and cost effective. Many surveys and books provided overviews of the existing solutions for data center interconnections such as [26], [8], [27], [28] and [29]. In the survey [26] and the book [8], the authors examined some representatives of data centers that appeared in the academic and research literature. Then, they analyzed the data center topologies according to several features and properties (scalability, performance, fault tolerance, ...). In addition, they studied the routing algorithms designed for each topology and the techniques used to enhance the data center

performance. However, recently designed data centers are not described in the aforementioned works. Moreover, the survey does not cover the wireless networks and the green cloud solutions. Authors of the survey [27] described a more recent data center architectures and discussed many novel techniques and schemes (transport protocols, sharing mechanisms, ...). Although this survey covered recent data center architectures and algorithms, there are still challenging issues like saving the energy consumption and minimizing the cabling/installation complexity which are not addressed in the paper. The authors of the surveys [28] and [29] focused on greening data center networks and described different techniques designed to minimize the consumed energy. In our work:

- We recall the well known and representatives data center network topologies (DCell, BCube, Fat-tree,...) described in the above surveys.
- We present new solutions that are not covered in all the previous surveys (SprintNet, Diamond, ...).
- We establish a comparison between these networks by plotting graphs of their performance parameters.
- We investigate the wireless networks and benchmark different contributions.
- We classify the contributions to green the data center networks, describe each type and present examples.

Data centers are classified as described in Figure 2.4.

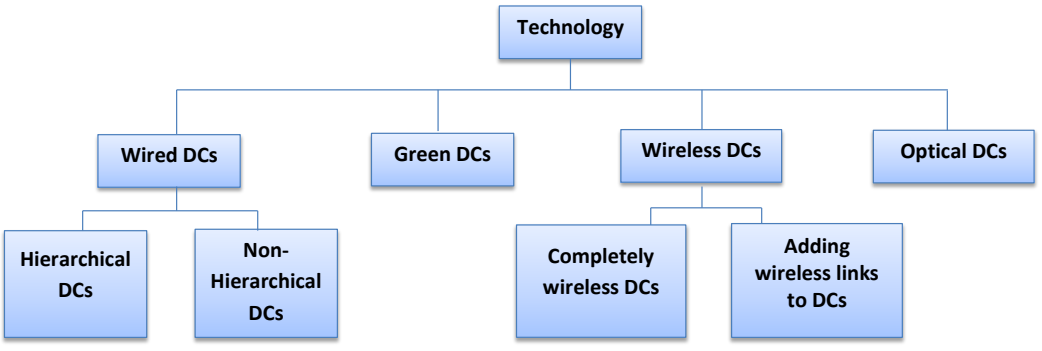


FIGURE 2.4: Classification of architectures.

2.5.1 Wired data centers

2.5.1.1 Hierarchical designs

The hierarchical designs build a network from multiple layers.

- Tree-based architectures

The tree-like topologies or Clos topologies are constructed as a multi-rooted tree.

- Fat-tree[11] is a 3-level data center. Using n-port switches, Fat-tree has n pods (see Figure 2.5). Each pod contains $\frac{n}{2}$ switches of aggregation layer and $\frac{n}{2}$ switches of edge layer. Switches in this topology are divided into $(\frac{n}{2})^2$ core switches and $\frac{n^2}{2}$ aggregation and edge switches respectively. The total number of servers hosted by the network is $\frac{n^3}{2}$. Even though Fat-tree is highlighted by its 1:1 oversubscription ratio provided to all servers, it still suffers from wiring complexity problem.

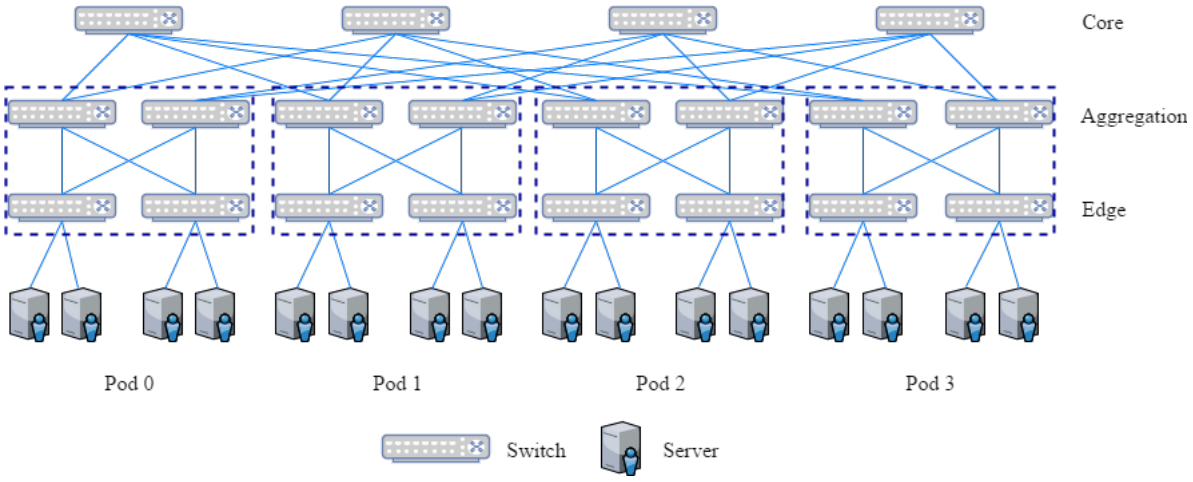


FIGURE 2.5: Fat-tree topology.

- VL2[30] is constructed of multiple switches arranged into a Clos topology as presented in Figure 2.6. The goal of VL2 is to solve the problem of oversubscription. In fact, using its algorithm of load balancing (Valiant load balancing algorithm) and its methodology to spread independent traffic to all destinations, VL2 offers a uniform high capacity to all servers. However, VL2 suffers from low scalability and expensive switches to be implemented (ports capacities are 10 times those of Fat-tree).

- Diamond[31] is an improvement of Fat-tree topology. But, it implements only core and edge n-port switches. In fact, Diamond supports $\frac{n^2}{2}$ edge switches connected to $\frac{n}{2}$ core switches. Each n edge switches connect directly Diamond servers. Compared

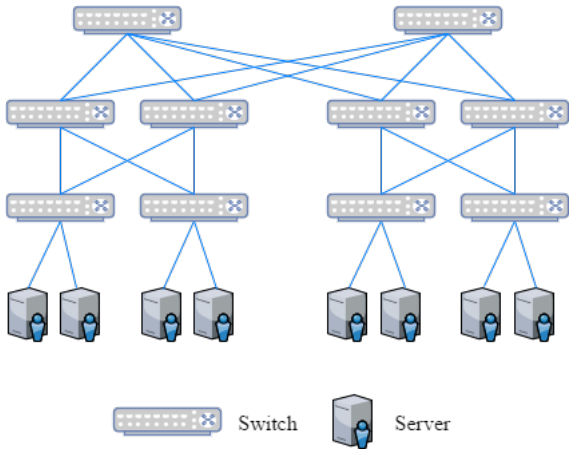


FIGURE 2.6: VL2 topology.

to Fat-tree, Diamond reduces the average path length while supporting the same number of servers.

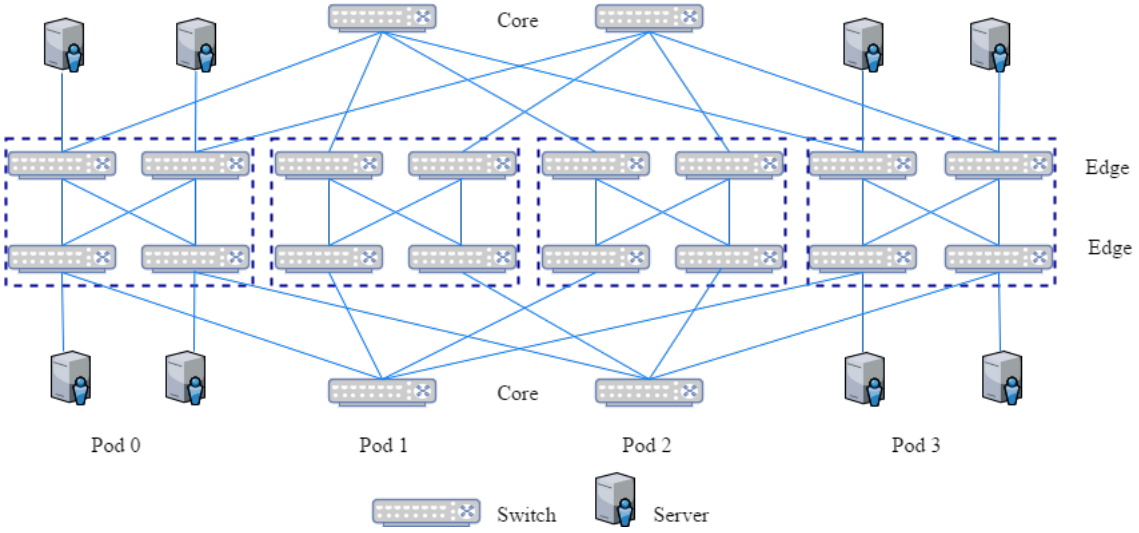


FIGURE 2.7: Diamond topology.

- Recursive architectures

The recursive architectures use servers with more than 2 ports to scale to k upper layers.

- DCell[9] is a recursive architecture, which is constructed from a low-end mini switch connected to n servers forming a basic unit called DCell_0 as depicted in Figure 2.8. To build a larger network, DCell_0s are connected with direct links between servers. DCell_k can be constructed from $(t_{k-1}+1)DCell_{k-1}$ where t_{k-1} is the number of servers in DCell_{k-1}. DCell offers a robust fault-tolerant routing protocol. However, despite its efficiency, lower levels burdened by more traffic suffer from bottleneck

and cause a low aggregate bottleneck throughput. In addition, the double exponential scalability is still the major challenge for DCell architecture.

- FiConn[32] has the same first layer as DCell_0 (see Figure 2.9). However, to scale to upper layers, FiConn uses the backup ports of the servers. FiConn_k consists of $(\frac{c}{2}+1)$ FiConn_{k-1} , where c is the number of backup ports in FiConn_{k-1}. The number of servers in FiConn_k is equal to $t_{k-1}(\frac{t_{k-1}}{2^k}+1)$, where t_{k-1} is the number of servers in FiConn_{k-1}. FiConn is a traffic aware topology that enhances the link utilization and the aggregate throughput. However, it suffers from bad fault tolerance and a long path length.

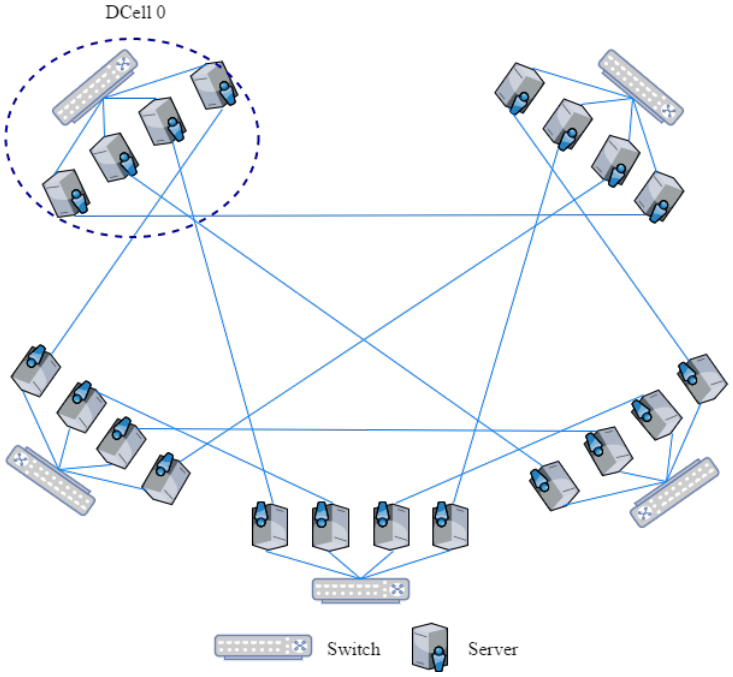


FIGURE 2.8: DCell topology.

- BCube[12] is also a recursive architecture designed for modular data centers, which makes it portable and easy to deploy. BCube has the same basic unit as DCell, however to construct a higher level network, servers are connected using n extra switches (see Figure 2.10). Consequently, a BCube_k is constructed from n BCube_{k-1} and n^k extra switches that connect one server from each BCube_{k-1}. With this huge number of network equipment and wires to enlarge the network, BCube cannot easily scale.

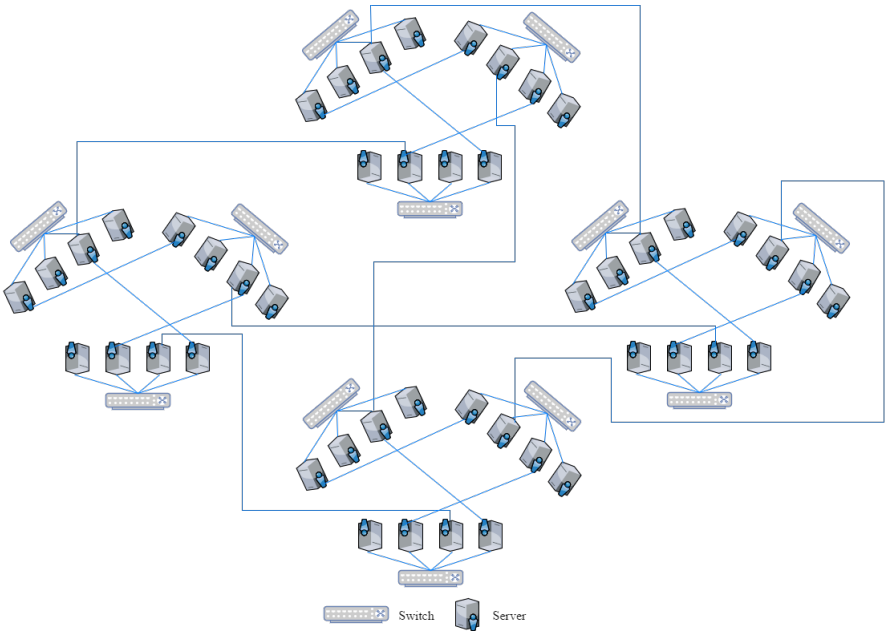


FIGURE 2.9: FiConn topology.

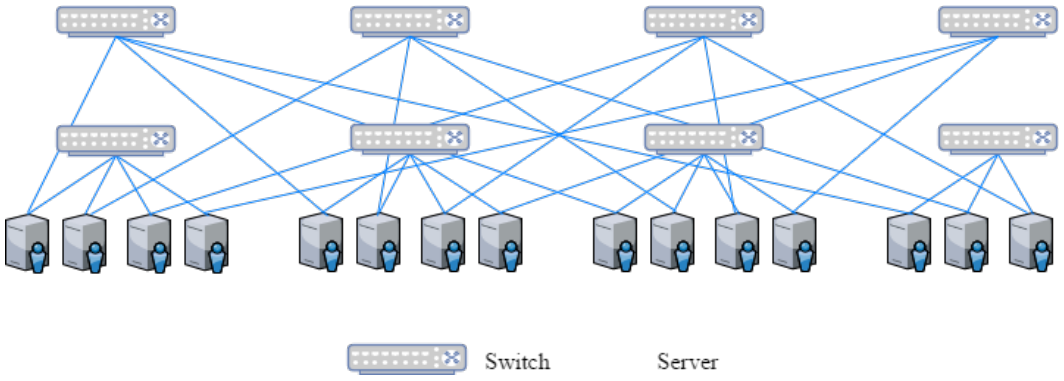


FIGURE 2.10: BCube topology.

2.5.1.2 Non-hierarchical designs

- Flat architectures

The flat architectures reduce the multiple switch layers to only two or one layer. The advantage of flat topologies is the easy management and maintenance.

- Flatnet[33] presented in Figure 2.11 is a flat architecture that scales with a speed of n^3. Its network offers a simple and scalable architecture, consisting of two layers. The first layer contains one n-port switch connected to n servers (1-layer Flatnet). The second layer is built by n^2 1-layer Flatnet (subsystems). Subsystems are connected using n^2 n-port switches. Flatnet shows good performance in terms of scalability

and fault-tolerance, however it does have some drawbacks, such as high diameter and long average path length that affect the latency of the network.

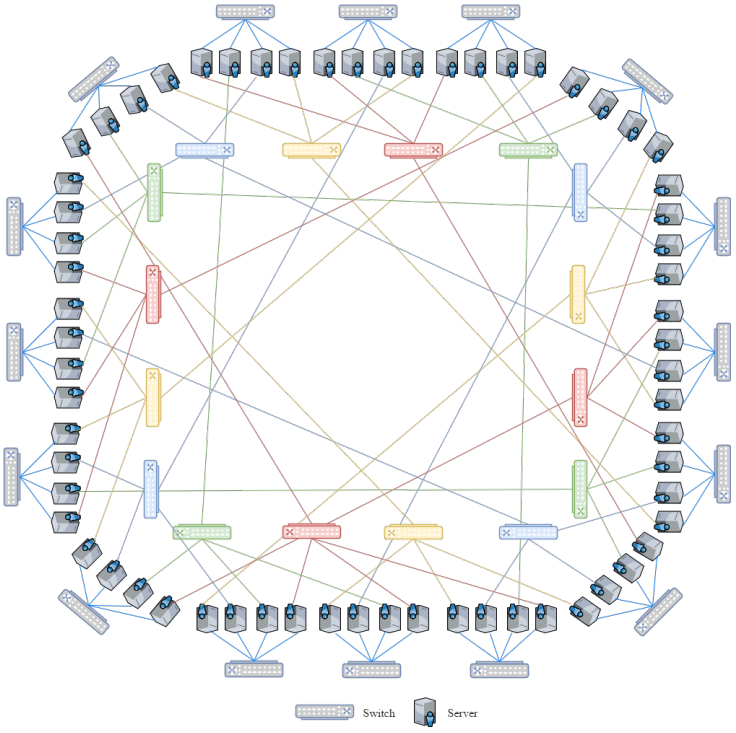


FIGURE 2.11: Flatnet topology.

- SprintNet[34] is a server-centric data center network architecture. This topology exhibits good fault tolerance and a low latency. The basic unit is named Cell, which is the building block to construct a larger SprintNet. Each Cell is constructed with c n-port switches, where $\frac{cn}{c+1}$ ports of each switch are connected to $\frac{cn}{c+1}$ servers and $\frac{cn}{c+1}$ ports of inter-Cell connections. Accordingly, each Cell contains $\frac{cn}{c+1}$ servers (see Figure 2.12). All the switches and servers are fully-connected. The 1-layer SprintNet consists of $\frac{cn}{c+1}+1$ Cells, and supports $(\frac{cs}{c+1})^2 n^2 + \frac{cn}{c+1}$ servers in total. A larger SprintNet can be constructed by adding $\frac{cn}{c+1}$ Cells fully connected to each others. Despite its low latency, SprintNet fails to construct a scalable topology.

- Switchless topologies

The switchless architectures does not rely on switches to build its network.

- CamCube[35] is designed to simplify the network topology. CamCube is a C-ary 3-cube, where C is the number of nodes in each dimension (see Figure 2.13). Each server in this topology connects directly 6 neighbor servers and can be defined by its coordinate (x, y, z) called address. CamCube can support up to C^3 servers. It

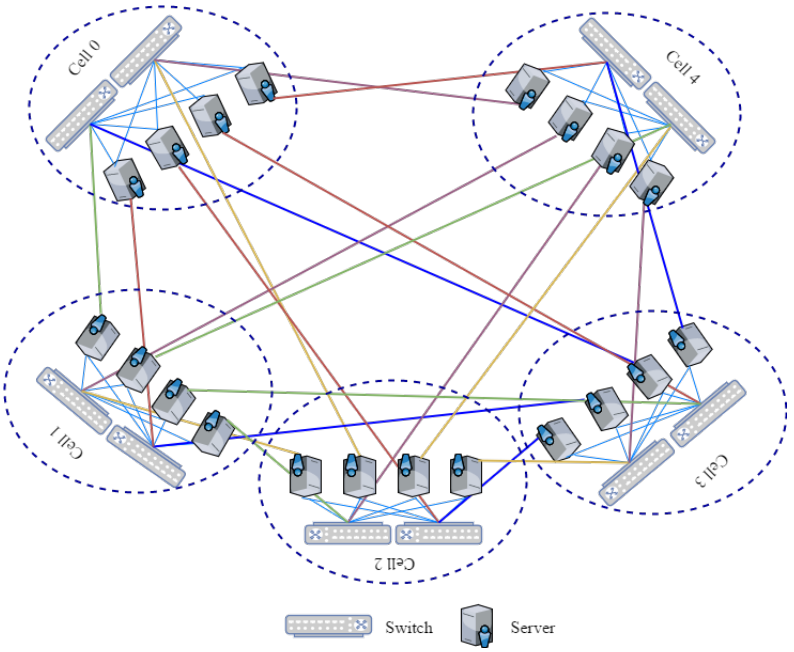


FIGURE 2.12: SprintNet topology.

is a scalable data center that offers multiple paths between communicating servers. However, these paths are not disjoint and the various shared links create congestion and packet loss, which can decrease end-to-end throughput.

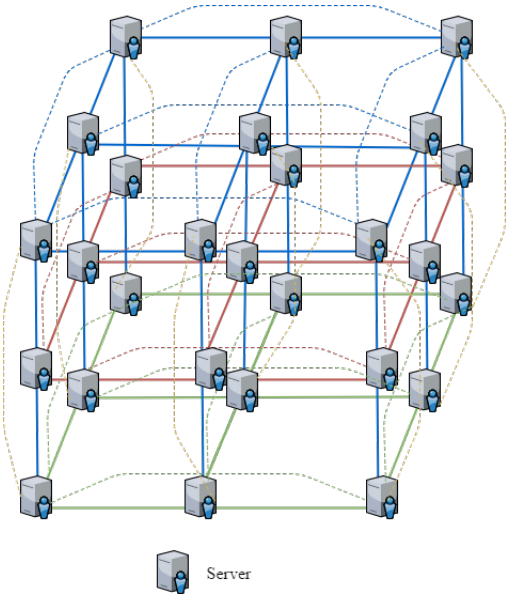


FIGURE 2.13: CamCube topology.

2.5.2 Wireless designs

2.5.2.1 Completely wireless data centers

The goal is to get rid of wires and offer a network with a reduced cost, facility of installation, re-configurability and easy maintenance.

- The polygonally arranged wireless data center is proposed by Vardhan et al.[36][2]. The goal is to replace all cables in data center networks by wireless links using IEEE802.15.3c standard known as 60 GHz millimeter wave technology (mmWave). Due to the actual disposition of data centers, servers in the same rack are not able to communicate with each others and farther servers in the same row cannot also communicate using wireless links. Authors proved that by placing racks in studied positions forming a polygon (hexagon for example) and by fixing the beamforming parameters, each node can communicate with other nodes. Figure 2.14 shows this hexagonal arrangement. Although this work came with a solution to create a

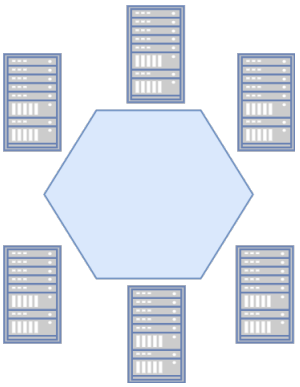


FIGURE 2.14: Hexagonal arrangement.

  topology free of wires, it can lead to an inefficiency of space usage and maybe cooling problems. In addition, no technical studies have been done to evaluate network performance (throughout, transmission delay...) after using wireless links.

- Caylay[1] is a novel topology designed to connect servers wirelessly. Authors suggested to build cylindrical racks. Every rack is divided into 5 levels called stories. Each story holds 20 prism shaped containers that stores servers. Network interface cards (NIC) of a server are replaced with a customized piece of hardware that connects the servers system with two transceivers placed at the ends of the prism. Racks are placed in rows and columns to make maintenance easier. Figure 2.15 shows this novel topology. Cayley owns a better bandwidth compared to Fat-tree, a good failure resilience and an acceptable cost and power consumption as maintenance costs and power consumption are significantly lower now due to the absence of wiring.

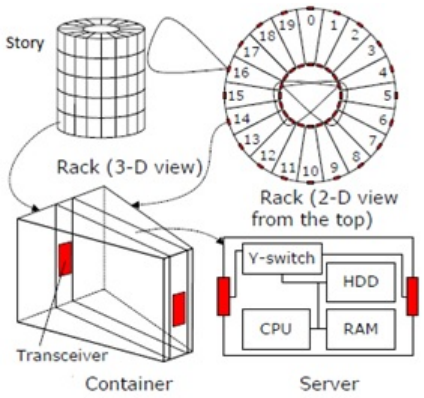


FIGURE 2.15: Cayley wireless Data Center[1].

However, Cayley presents some limitations such as packet delivery latency and scalability because of the multi hop nature of the topology.

2.5.2.2 Adding wireless links to data centers

Traditional data centers are tree-structured topologies. This type of architecture suffers from oversubscription in Top of Racks (ToR) switches (used in full capacity, congested). Each oversubscribed switch that presents a potential hotspot can block some applications and deteriorate the performance. Since implementing more links and switches to solve this problem can be costly, recent researches propose to add wireless links in ToR switches which is a less expensive solution.

- Srikanth et al. [37] studied traffics in a tree-based data center and tried to gather demands between top of rack switches. They conclude that traffic is concentrated between a few ToR switches that present a bottleneck and hold back the entire network from job completion. Thus, providing extra capacity for these ToRs can significantly improve performance. In fact, a network with few wireless links (flyways) connecting pairs of oversubscribed ToRs with a low bandwidth can offer a performance equal to the one of a non-oversubscribed network. Furthermore, flyways architecture is not coming with a new structure like VL2 and Fat-tree. Rather, wireless links can be deployed anytime on tree-topologies.

- 3D beam-forming for wireless data centers[38][39] is designed after studying previous works aiming to increase performance by adding wireless links. In fact, restricting wireless transmissions to neighboring racks means that multiple hops connection is needed to connect two non-line of sight nodes. This can increase the transmission delay and reduce throughput. In addition, due to the interference between different wireless links, concurrent transmissions are restricted. Thus, authors proposed

to use 3D beam-forming. This approach leverages ceiling reflections to connect non neighboring racks as illustrated in Figure 2.16. Indeed, each sender points its beam to the ceiling, which will reflect the signal to the desired receiver. 3D beam-forming approach requires beam-forming radios, ceiling reflectors that act like specular mirrors to reflect signals and electromagnetic absorbers placed on the top of the racks to prevent any local reflection and scatter around the receiving antenna. Ceiling reflection can extend link connectivity to reach far nodes and bypass obstacles. In addition, 3D beam-forming limits the interference region to a much smaller area. Experimental results show that the majority of the rack pairs can be connected simultaneously. However, many challenges face this work. Actually, the rotation to the ceiling and the positioning of the wireless link add an extra time which can slow the transmission delay. Finally, in some data centers, ceiling can be used to house pipes and cables. So, it will not be easy to install reflectors.

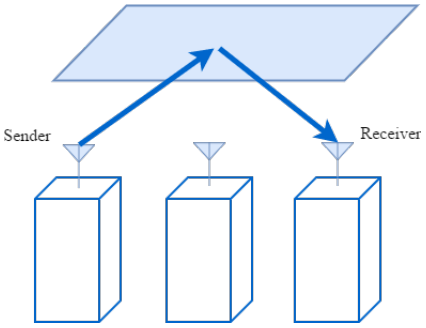


FIGURE 2.16: 3D beam-forming.

2.5.3 Optical designs

- C-Through (HyPaC)[40] is a topology that combines a traditional tree-based network and an optical circuit network to connect the ToR switches. C-Through is composed of two components: control plane and data plane. The control plane estimates the traffic between racks and defines the new circuit to send the load. The data plane isolates the optical and the Ethernet networks and de-multiplexes the load into the calculated routes. The optical routes have the priority over the electrical routes because they offer a higher capacity to the data center. However, optical links can be costly to the network.

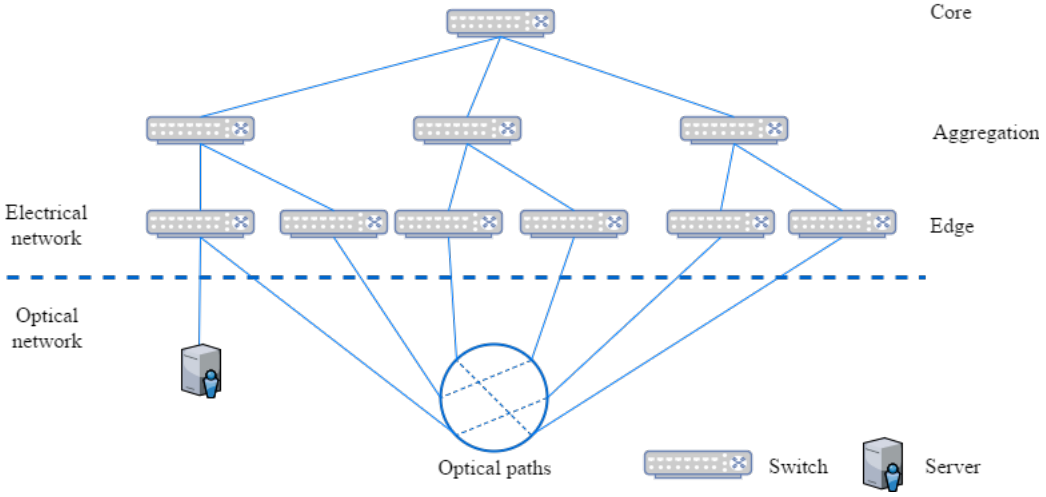


FIGURE 2.17: Hypac topology.

2.5.4 Green data centers

Minimizing the power consumption of the network equipment is the target of all network operators whatever is the type of their systems: Radio networks[41] , sensor networks[42][43], backbone networks[44], data center networks[45],... In this context, many efforts to build a green data center have been conducted recently in both industry and academia. A short review of the most promising approaches in this field is summarized as follows:

2.5.4.1 Usage of renewable sources of energy

This approach exploits the green resources such as water, wind, solar energy, pumps and heat to reduce the budget related to energy. For example, authors in [46] proposed to make all data center devices almost powered entirely by the renewable energy. To realize a testbed, they added temperature and humidity sensors, solar powered systems and wind power systems to a data center. Using this method, they obtained a successful results and witnessed the launch of several international collaborations including the US Greenlight and the New York State Energy Research. Another proposal to introduce the green energy in DCs is the net-zero networks[47]. This new concept consists of producing an amount of energy per day that is equal to the same amount of energy consumed by the network. The renewable energy is used nowadays by big companies such as Google[48] and Facebook[49]. However, the renewable energy sources used to power the network are limited by several factors such as the weather conditions and the location of the data center, in addition to the high cost of the infrastructure to generate the power and deliver it.

2.5.4.2 Energy aware hardware

This method focuses on making the energy consumed by switches and servers proportional to the traffic load. To achieve this, some specific techniques are applied such as vary-on/vary-off (VOVO)[45]. Workloads in VOVO are concentrated in a subset of servers instead of distributing it across all servers which ensures more energy efficiency. Another technique is the Dynamic Voltage (DVFS)[50] where the CPU power is adjusted depending on the arriving load. The idea is based on the fact that the power in the chip is proportional to $V^2.f$, where V is the voltage level and f is the frequency. This ensures that the power consumption of a chip is proportional to the workload. However, these approaches optimize only the energy consumed by the CPU while the remaining network devices are untouched and are still working on their normal energy level.

2.5.4.3 Energy aware architecture

This relates to the design of a data center that conserves energy thanks to its architecture. CamCube[35] is designed to interconnect servers directly without intermediate switches (switchless networks, see section 2.5.1.2). This type of topologies saves energy consumed by switches, racks and associated cooling machines. Another example is the Nano data centers[51], where services and content are stored in the home gateways instead of being stored in data centers. To access to the content in the gateways, a P2P infrastructure is used. This approach can only be applied to a small sized network and needs more investigation to be feasible in a real data center network. Wireless data center topologies[36][37] are also an attempt to minimize the energy consumption. In fact, by relying on transceivers with a minor energy consumption compared to switch interfaces[1], communications are delivered without wasting a huge amount of energy.

2.5.4.4 Usage of virtual machines technology

The virtualization technology is based on creating multiple virtual machine (VM) instances on a single physical server[52][53]. VM can act like a real machine with an operating system. Thus, by powering only one device while multiple machines are processing, the amount of hardware in use is reduced, the resource utilisation is improved and the energy consumed is optimized. Recently, virtualization tools are available to test and use by some vendors such as VMware[54]. GreenCloud[55] is one of the optimisations used for migration and placement of virtual machines in data centers. In this project, the authors studied the workloads of the applications in cloud networks.

Depending on the workloads, they proposed algorithms for mixing and mapping virtual machines to the existing resources while considering the QoS (Quality of Services) requirements and energy efficiency. The GreenCloud approach consists of two phases. The first one deals with the collection of new demands and the second one is to optimize the allocation of the VM on physical machines depending on the demands. Then, idle machines are turned off.

2.5.4.5 Routing level power aware approaches

This type of approaches aims at saving the energy consumed by the idle devices that are not included in the traffic but are still wasting energy. In fact, it is based on a power aware algorithm that consolidates the traffic flows, restricts the network to a subset of devices and powers off the idle ones. These approaches are applied at first to a general network. Among these contributions, the following works proved their efficiency to reduce the energy consumption by switching off the idle nodes.

- The authors of [44] proposed a greedy heuristic to switch off nodes and links in a backbone network in order to minimize its power consumption. In this way, during the low load hours, the small traffic can be routed on a defined subset of the network and a large number of nodes can be switched off while respecting connectivity and QoS constraints. The heuristic starts by considering that all elements of the network are powered on. Then, it iterates on network elements to define which ones to switch off. At each step, the traffic passing by the candidate node to power off is rerouted using a calculated shortest path and the links utilization are checked to respect the QoS thresholds. If the constraints are not violated, the selected nodes/links are switched off. Several policies such as random (R), least-link (LL), least-flow (LF) and opt-edge (OE) policies can be adopted to sort the node set and reduce the iteration complexity. The R sorts the nodes randomly. Then, LL re-sorts the nodes according to the number of links connected to them. The node with the smaller number of links comes first. In the next step, the LF places first the nodes that have the smallest amount of load passing by them. Finally, the OE keeps the nodes source and destination so that they cannot be powered off. But, the links of these nodes can be deactivated as long as one of them is active. The proposed heuristic proved its efficiency on reducing the power consumed by a backbone network. However, this method can reduce aggressively the redundancy of the network and in case of failure, the communication can not instantly be rerouted to other paths. In addition, even though the heuristic reduces the complexity of choosing the set of nodes to power off, searching the shortest and available paths for all the flows is still a complex task that takes time and can enlarge the latency of the communications.

- Authors of [56] propose a new ECO-friendly routing algorithm named ECO-RP based on the open shortest path first protocol (OSPF)[57]. OSPF is a well-known routing protocol that calculates shortest paths based on links weights assigned by the network operator. These weights are generally fixed. It means changing the routes according to the traffic loads is not considered by the OSPF protocol. In the new proposed protocol, new entities called ECO-RP are introduced in each network device. Their role is to check the traffic forwarded by the device and send the information to the other entities. In this way, every ECO-RP has an overview about the network traffic and can change OSPF link weights according to the traffic status. If the traffic is low, the entity changes the links weights so that traffic is rerouted through other paths and a subset of the network can be powered off. Hence, unlike OSPF, the new protocol changes the network routing based on the network traffic. In addition, in the standards of OSPF, the sleeping nodes are considered as failures and their information are removed. However, ECO-RP can distinguish between failed and sleeping nodes. To manage the ECO-RP protocol capabilities, many messages and databases are used including Network State Advertisement (NSA) that checks the traffic periodically, Network State Database (NSDB) that collects the traffic status from all entities and Historical Link State Database (HLSDB) that distinguishes between failures and sleeping nodes. The new protocol proved its efficiency to reduce energy consumption while maintaining the performance of the network. However, the energy consumption is not very big reaching at maximum 18%.

- The authors of [58] propose a distributed routing protocol named Distributed Routing Protocol for Power Saving (GDRP-PS) to save energy in the core networks by putting idle routers into sleep mode while maintaining the network connectivity and the QoS constraints. During the high loads, this algorithm is not applied so that the performance is not deteriorated. In the low loads, GDRP-PS can impose the sleeping mode on idle routers. In addition, this algorithm focuses only on the core routers because the edge routers are connected to final users and can send or receive data at any time. Also, a part of core routers called traditional routers will stay always powered on and will use the traditional network routing distribution (e.g. OSPF). The other set of routers, called power saving routers (PSR), can switch from working to sleeping mode. The GDRP-PS coordinator detects the status of the network and impose the mode of the PSRs. In fact, each PSR checks if the network will still connected when it is sleeping. Then, it sends a message to the coordinator to ask for mode switching. If it can go to sleeping, it rebuilds its routing table and switches off for a certain period. If not, it waits for a fixed time and asks the coordinator again. While the PSR is sleeping, it can wake up again either if the period ends or if it receives a wake up message from the coordinator. GDRP-PS proved that it can

significantly reduce the energy consumption of the core network while maintaining the performance of the network. Moreover, it is designed to enable the co-existing of multiple routing protocols. However, the range of machines to power off is very reduced in such a way the energy gain is not very significant.

Such approaches are then applied to data centers. In fact, data centers are special networks. They are characterized by the regularity of their architectures which can help to design a less complex algorithm. Also, the data centers are huge networks that can host thousands of servers. Hence, a non complex and a time efficient algorithm is required to be applied to a large number of devices without degrading the performance. Among the contributions designed for data centers, we can cite:

- Elastic tree[13] is an approach that proposes a power aware strategy. This strategy aims at finding the minimum subset of the network that should stay active and shutting down the unused set of network devices. The Elastic tree consists of three modules as shown in Figure 2.18: The first one is the optimizer. Its role is to choose the minimal number of devices that contribute in the traffic communication. The second module is the routing part where routes are calculated. Finally, the power control module is responsible for adjusting the state of devices. Three optimizers are proposed to calculate the needed subset of the network which are the topology aware heuristic, the formal model and the greedy bin packing. The formal model targets the optimal power solution by searching the optimal flow assignment while satisfying all traffic constraints. Thus, finding the optimal flow assignment is an NP-hard problem and costs a long period of computation that can reach several hours. This optimizer suffers also from a poor scalability and can be applied on a maximum of 1000 network-devices. Although it has only a medium scalability, the greedy bin packing improves the scalability of the optimizer; but an optimal solution is not guaranteed. In fact, for every flow, it calculates the possible routes, chooses the left-most path with the best capacity and powers off the rest of the network devices. The topology aware heuristic is designed for the FatTree[11] topology since it uses the regularity of this architecture to quickly find the network subset and keep it active. This approach has the best scalability and the smallest computation time compared to the other two optimizers but it delivers the worst optimization quality since it does not compute the exact flows routes.

- The traffic merging[59] aims at reducing the energy consumed by switches. The idea is to consolidate the traffic arriving to a switch from different links and feed them to few ports of the device. Thus, by merging the traffic, the authors ensure that several interfaces are operating under low power mode. For example, if N flows are coming

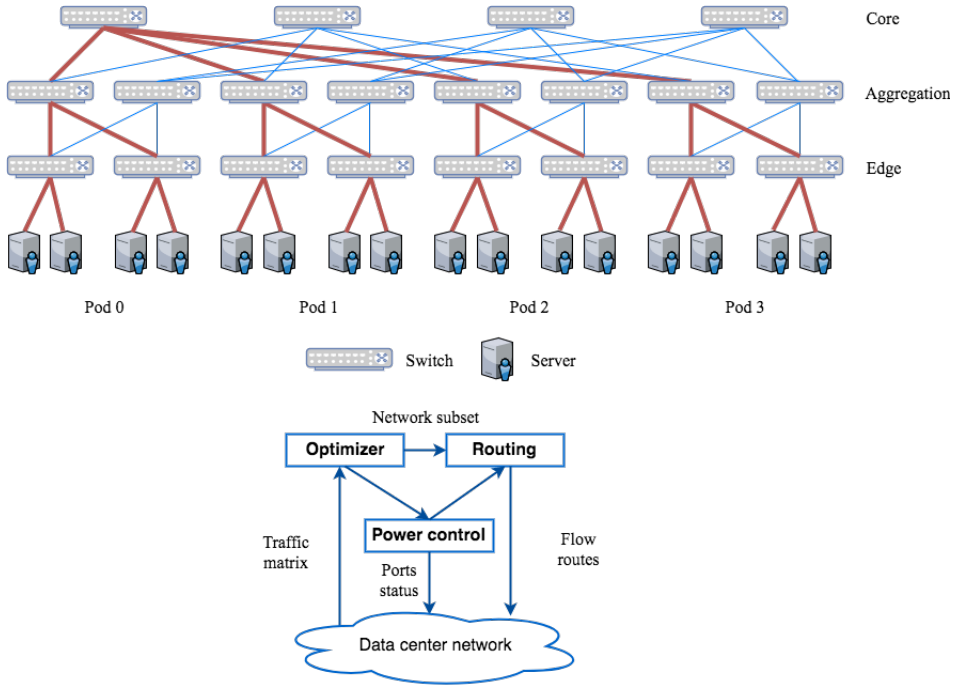


FIGURE 2.18: Elastic tree power aware topology.

from N links, this approach allows to merge them and feed them to K interfaces. Setting the parameter K depends on the arriving loads and allows to activate only K interfaces and put $N - K$ ports into low power. The traffic merging shows an optimal energy saving. However, it saves only the energy wasted by switches. Also, an unexpected failure in the merged link or the switch port can lead to a loss of multiple communications.

- The energy aware routing model[60] addresses the power saving problem from a routing perspective. The key idea is to use the minimal number of switches while providing the routing services and maintaining the targeted throughput. In fact, the authors propose to generate, first, the routes and compute the throughput of the network without powering off any switch. Second, they eliminate the unneeded switches, switch by switch until reaching a certain threshold. Finally, they power off these switches. This approach suffers from an expensive computation time due to the elimination of switches one by one and the recalculation of the near-optimal solution. Also, by powering off only switches and their related links, many unused ports in other switches and servers are still wasting energy. In addition, this algorithm does not consider the robustness of the system and the latency of the network which is important especially for a latency-sensitive data center network.

- The artificial intelligence abstraction approach[15] proposes an intelligent bandwidth allocation mechanism using the blocking island paradigm (BI) to achieve power conservation. Two phases are proposed. First, by applying the BI that provides

an efficient way to represent the availability of the network resources, the space of searching the routes for the communication flows is reduced and routing paths are calculated with a lower complexity. The second phase consists of the power aware heuristic routing scheme. In this module, the best set of routes to satisfy the traffic demands is chosen and the devices that are not impacted in these routes are powered off. Even though this approach decreases the complexity of calculation of routes for the traffic demand, it still has an expensive computation time to find all routes and filter them.

2.5.4.6 Queuing analysis for power aware approaches

Queuing theory[61] is a deeply established analyze that is well-known and well-suited to study the networks. This theory helps to predict the workloads, the performance change, the traffic volumes and the traffic scenarios. Few efforts use the queuing models in data centers including:

- The energy optimization on device interfaces approach[62] consists of changing the state of the port according to the queue length to adaptively adjust the energy consumed by switch ports according to the traffic load. In the initial stage, all switches are inactive. Switches are enabled only when packets arrive. Then, when β packets reach the interface of the switch, the queue is examined. If the buffer level decreases to T_{down}, the port downgrades to a lower rate. However, if the buffer level exceeds T_{up}, the port upgrades to a higher level. Each port may experience several states including Sleep state, 100Mbps, 1Gbps and 10Gbps of data rate.
- The task managing based on vacation M/G/1 queuing model[3] is an approach that models packets scheduling in an heterogeneous data center network using an M/G/1 queuing analyse. Nodes are normally running in high power. If there is no incoming packets and servers are switched to idle state, a low power level is set. Servers are activated when a job is present. Sejourn time of a packet in the system is calculated and proven to be acceptable while gaining a large amount of energy. Still, since the modeled queue has an unlimited length, the energy saving is not possible when the traffic load is high and the node has to be always active.
- The task managing based on vacation M/M/n queuing model[63] consists of proposing a threshold oriented model to reduce the energy consumed by servers in tree-based data centers. Specifically, the authors adjusted the number of active servers and turned off the others. The state of the nodes is decided by the packets arriving from the top rack switches. If the incoming jobs in the queue reach a certain threshold, some extra servers must be activated. The optimal trade-off between the power

saving and waiting time is determined by the M/M/n analytical model. Results show a large power saving amount with an acceptable waiting time. However, the waking up time taken by servers before activation is not considered.

2.6 Comparison of data center architectures

In this section, some typical features of different architectures are explored.

2.6.1 Comparison of wired designs

The analyses of the structural properties of wired architectures introduced earlier are summarized in Table 2.1, which also present a comparison among these networks. Many surveys such as [26], [27] and [21] have conducted a similar comparison. In our work, we included all our benchmarked topologies and we established a comparison by plotting the parameter graphs (diameter, scalability, wiring complexity, cost,...). These graphs are presented later in this section. N_S is the number of servers, N_{SW} is the number of switches, n is the number of ports per switch, k is the number of layers in the network, c is the number of switches per SprintNet cell and C is the number of nodes in each dimension of CamCube.

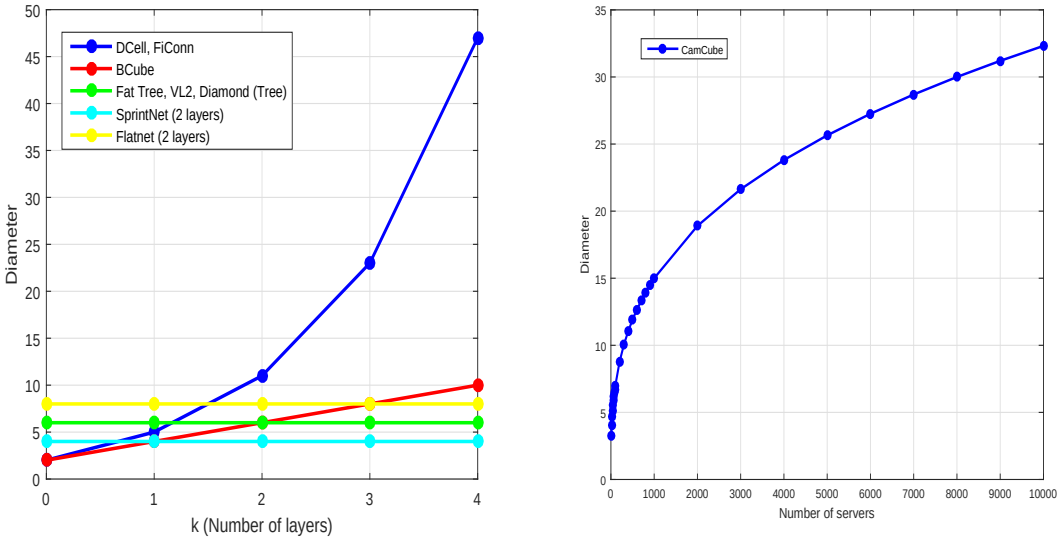
TABLE 2.1: Quantitative comparison of different data centers.

Classification	DCN	N_S	N_{SW}	Degree	Diameter	Bisection width
Tree-based	Fat Tree	$\frac{n^3}{4}$	$\frac{5n^2}{4}$	1	6	$\frac{N_S}{2}$
	VL2	$5n^2$	$\frac{n^2+6n}{4}$	1	6	$2N_S - 20$
	Diamond	$\frac{n^3}{4}$	$\frac{5n^2}{4}$	1	6	$\frac{N_S}{2}$
Recursive	DCell	$\in ((n+\frac{1}{2})^{2^k} - \frac{1}{2}, (n+1)^{2^k} - 1)$	$\frac{N_S}{n}$	$k+1$	$< 2^{k+1} - 1$	$> \frac{N_S}{4log_n N_S}$
	FiConn	$\geqslant (\frac{n}{4})^{2^k} 2^{k+2}$	$\frac{N_S}{n}$	2	$< 2^{k+1} - 1$	$> \frac{N_S}{2^{k+2}}$
	BCube	n^{k+1}	$(k+1)n^k$	$k+1$	$k+1$	$\frac{N_S}{2}$
Flat	Flatnet	n^3	$2n^2$	2	8	$\frac{n^3}{4}$
	SprintNet	$(\frac{c}{c+1})^2 n^2 + \frac{c}{c+1} n$	$\frac{c^2}{c+1} n + c$	$\geqslant 2$	4	$\frac{c^2 n^2}{2(c+1)^2} + cn$
Switchless	CamCube	C^3	0	6	$\frac{3}{2}\sqrt[3]{N_s}$	$8\sqrt[2]{N_s}$

2.6.1.1 Comparison of network diameter

By comparing different architecture proposals as described in Table 2.1 and Figure 2.19, VL2, Fat Tree, FlatNet, BCube and sprintNet outperform the other topologies thanks

to their low diameters. The network diameters of the other architectures (DCell and CamCube) are higher and increase as the number of servers/layers.



(a) (b)

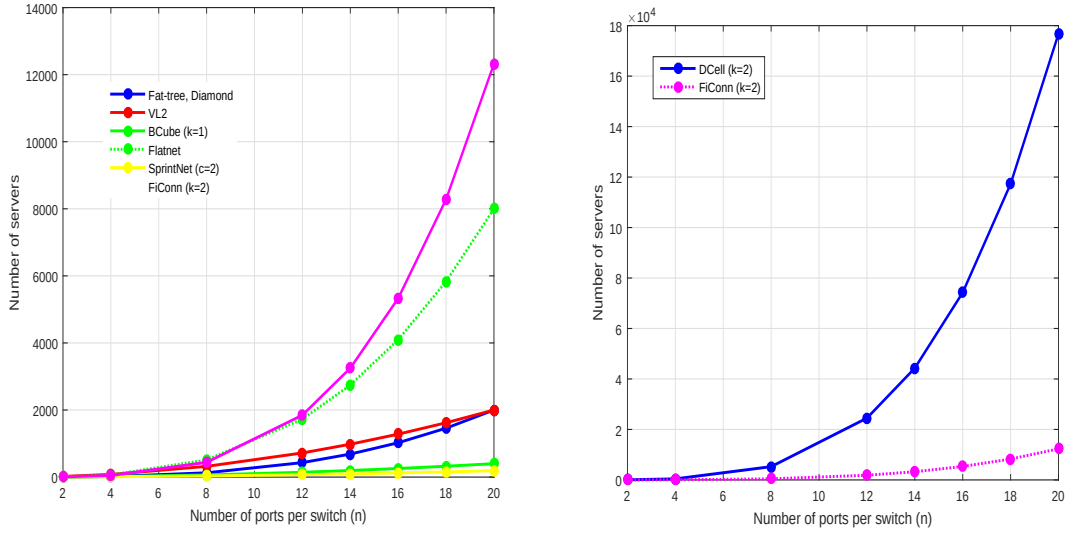
FIGURE 2.19: Comparison of Diameter among different topologies.

2.6.1.2 Comparison of scalability

A 2-layer FlatNet, Fat Tree and Diamond have an $O(n^3)$ scalability which is higher than a 2-layer DCell, FiConn, BCube, VL2 and SprintNet of $O(n^2)$ scalability as indicated in Table 2.2. However, as the number of layers increases, DCell and FiConn own the larger scalability as indicated in Figure 2.20.

TABLE 2.2: Comparison of scalability.

Architecture	Scalability	
	2-layers	k-layers
Fat-Tree	$O(n^3)$	–
VL2	$O(n^2)$	–
Diamond	$O(n^3)$	–
DCell	$O(n^2)$	$O(n^{2^{k-1}})$
FiConn	$O(n^2)$	$O(n^{2^{k-1}})$
BCube	$O(n^2)$	$O(n^k)$
Flatnet	$O(n^3)$	–
SprintNet	$O(n^2)$	–



(a) (b)

FIGURE 2.20: Comparison of scalability.

2.6.1.3 Comparison of cost

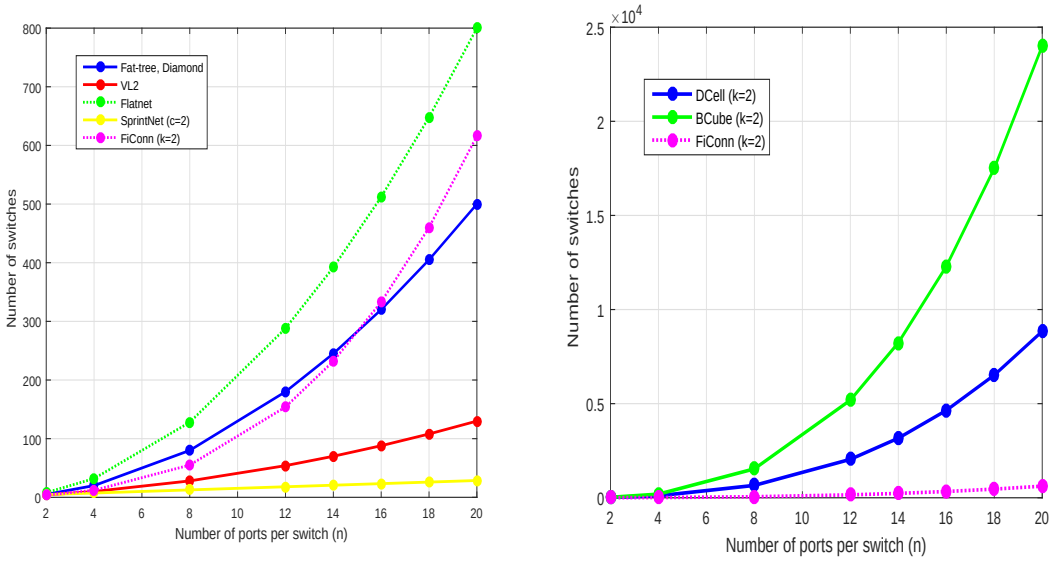
The cost of the network depends of multiple factors including the number of switches, networking devices, number and type of cables and power consumption. Figure 2.21 presents the number of switches among different topologies. It can give an idea about the cost of the data center netwrok. The architectures DCell, BCube, FlatNet and FiConn outweigh the other topologies as they are implementing more networking devices.

2.6.1.4 Comparison of bisection width

Figure 2.22 presents the bisection width of different topologies. As we can see, 3-layer DCell, 3-layer BCube and VL2 outperforms other topologies in terms of bisection width.

2.6.1.5 Comparison of cabling complexity

Figure 2.23 presents the cabling complexity of different topologies. As we can see, the 3-layer DCell, 3-layer BCube and VL2 deploy more cables to build their topologies than the other designs.



(a) (b)

FIGURE 2.21: Comparison of number of switches.

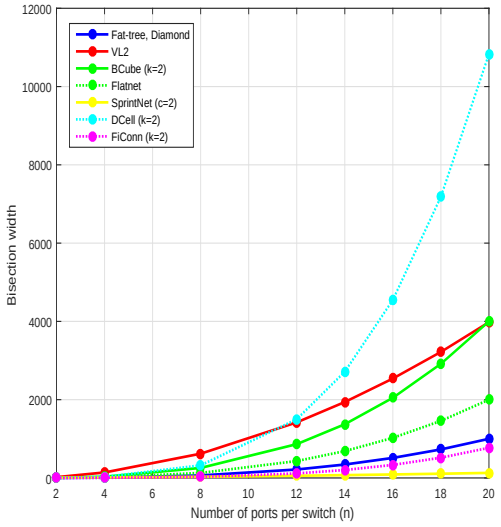


FIGURE 2.22: Comparison of bisection width.

2.6.2 Comparison of wireless designs

Wireless networking can be a possible solution to handle the limitations of wired network architectures such as wiring complexity, maintenance and re-configurability. After conducting a review of research efforts related to the wireless DCs field[64], a comparison between wireless DC networks is presented in Table 2.3 and some remarks are stated as follows:

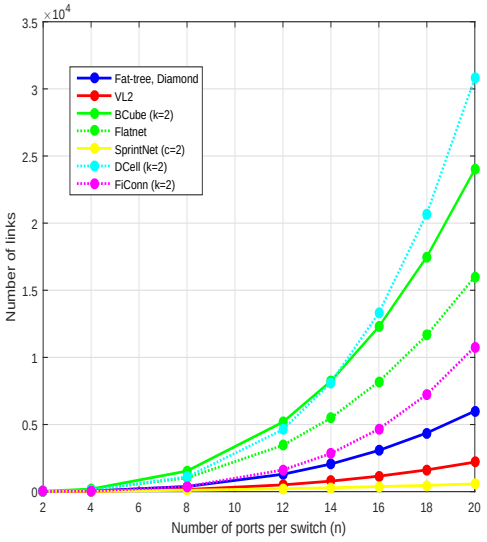


FIGURE 2.23: Comparison of number of links.

TABLE 2.3: Comparison between wireless DCs.

Category	Advantages	Drawbacks
Replacing cables with wireless links in existing DC topologies.	-Removing wiring complexity. - Ease of maintenance. - Ease of installation. - Re-configurability. - Reduced cost and power consumption. - The novel topology does not use switches that present a critical point of failure.	- Inefficiency of space usage and cooling mechanisms. - Wireless links alone can not meet the performance of wired links.
Creating a novel wireless DC topology.		- Packet delivery latency. - Scalability.
Constructing an hybrid DC.	- Reduce congestion in hot nodes. - Solve the problem of oversubscription.	- Complexity of channel allocation for wireless transmissions.

- Performance: A wireless data center architecture must satisfy basic requirements such as high capacity, fault-tolerance, and scalability. However, it is difficult for a completely wireless data center network alone to achieve all requirements. In fact, the speed of wired links in data centers continues to increase while the maximum rate of wireless links is 7 GBps which can be attenuated by interference. Another concern can be raised:

Multi-hops and non-line of sight which make wireless networks suffer from a big latency and low scalability.

- Containerized data center networks: Most of nowadays data centers use a large open space plan but some of them are being built with containerized architecture. In this case, wireless links can suffer from multi-path effects and communication between containers can be blocked by walls.

- Power consumption: According to [65], the maximum consumption of a 60 GHz transceiver is 0.3 watts. For two transceivers in 10K servers, the total power consumption will be 6 kilowatts. However, wired switches can consume 72 Kilowatts in 10K servers data center which favors the use of wireless links.

- Hardware cost: It is difficult to calculate the cost of a wireless network because there is no commercial vendor of 60 GHz transceivers and just some samples have been produced. However, authors of [2] estimated the cost of wireless and wired data centers and conducted a comparison as shown in Table 2.4. Let $C_{TX}, C_{WDCN}, C_{Fat-Tree}, C_{switch(k)}$ and k be respectively the maximum cost of a transceiver, the cost of a wireless data center, the cost of a Fat-tree topology, the cost of a switch with k ports and the pod size of a Fat-tree. This comparison shows that there is no big difference between wired and wireless topologies. So, maybe the cost of installation and maintenance favors the choice of using 60 GHz technology in data centers.

TABLE 2.4: Cost comparison of wired and wireless data centers ($)[2].

K	$C_{switch(k)}$	$C_{Fat-Tree}$	C_{WDCN}	C_{TX}(estimated)
16	2000	640k	4096 C_{TX}	156
24	4000	2.8M	6912 C_{TX}	405
32	6000	7.68M	16.3864 C_{TX}	468
48	7500	21.6M	55.296 C_{TX}	390
96	70.000	806.4M	442.368 C_{TX}	1822

2.7 Conclusion

Data centers are becoming more and more popular for wide variety of applications. However, more challenges and issues related to their networks are appearing and need urgent solutions. Thus, many researches have been conducted in this topic. In this chapter, we have reported a comprehensive review of the works done for data centers networking. We have described and discussed the proposed approaches and contributions. Although these contributions improve the network performance and address

many issues, other challenges remain to be handled. By understanding these challenges, we came with new alternatives that will be described in the next chapters.

Chapter 3

Improving the QoS of data center networks

3.1 Introduction

The data center network connecting thousands of servers plays a crucial role to deliver peak performance services with strong reliability to users. As noted in the previous chapters, this large-scale data center serves as the core infrastructure for the Cloud. To support the growing cloud computing needs, the number of servers is increasing exponentially, thus resulting in enormous challenges to create an efficient network design with a simple deployment, maintenance, a low cost and a high performance. Designing such a data center architecture network that satisfies all these requirements has recently become a hot topic in the research community. Many solutions have been proposed but they often fall short of practicality either because they suffer from a slow/double exponential scalability or bottleneck problems alike wiring complexity, cost of construction and high diameter.

In this chapter, we try to design two data center interconnections that solve essentially two of the most important challenges: latency and scalability. Meanwhile, the new designs enjoy low cost, power efficiency and high capacity. The main contributions in this chapter can be summarized as follows: (i) The investigation of the topological and physical characteristics of the two new data center architectures. (ii) The theoretical analysis and practical simulations we have conducted to evaluate their performances.

3.2 Improving the latency: wireless technology (wFlatnet)

In this section, we are improving the performance of an existing data center network topology (DCN) in terms of latency and average path length taking into consideration the load balancing and fault tolerance properties. An intuitive idea would be to add some switches and links that may reduce the path length at a price of high complexity of wiring and installation in addition to the cost of cabling which is very high . So, we propose to add wireless links as a solution to reduce the transmission path. The new topology, called wFlatnet[66], introduces shortcuts between nodes through wireless connections without burdening the network with wiring and maintenance complexity.

3.2.1 Motivation

3.2.1.1 Flatnet: A long average path length

After conducting a study of different data center architectures in chapter 2 to understand encountered problems and proposed solutions, the investigation reveals that Flatnet[33] is a promising data center architecture that presents excellent performance (scalability and fault tolerance) compared to widely known data center networks such as DCell [9], BCube[12], VL2[30], etc. Flatnet combines the advantages of previous architectures while avoiding their limitations. However, this topology suffers from a long path length problem which affects the latency of the network. In fact, Flatnet possesses the biggest diameter as shown in Table 3.1. Packets in Flatnet are transmitted using routing paths of length 2, 4, 6 or 8. But, in All-to-All communication pattern, the most of routing paths are of length 6 and 8 and the percentage of 6 and 8 routing paths increases when augmenting n (number of ports per switch), *e.g.* for n=8 and n=16, this percentage is equal to 78.08% and 88.27% respectively. So, the average path length will be affected and will consequently impact packet transmission delay and the completion time of applications.

This issue has motivated us to propose a high performance data center based on Flatnet with some enhancements aiming at solving the problem of long path length and high network latency. The first idea was to add links and switches to construct shortcuts between servers but a simple investigation of wiring complexity in different topologies reveals that Flatnet equipment are connected with a huge number of cables. In fact, Figure 3.1 illustrates a comparison of number of links among different data center architectures and shows that Flatnet exhibits the largest number of links. Therefore,

TABLE 3.1: Properties of different network architectures.

	VL2 (3 layers)	DCell (2 layers)	BCube (2 layers)	Flatnet (2 layers)
Diameter	6	5	4	8
servers number	$\frac{(n-2)n^2}{4}$	$n(n+1)$	n^2	n^3
Links number	$\frac{(n+2)n^2}{4}$	$\frac{3n(n+2)}{2}$	$2n^2$	$2n^3$
Switches number	$\frac{3n}{2}+\frac{n^2}{4}$	$n+1$	$2n$	$2n^2$

adding more links will only intensify the burden of cabling which makes the installation and the maintenance of the network more complex. This encouraged us to think about the wireless technology that proved its efficiency in data center networks.

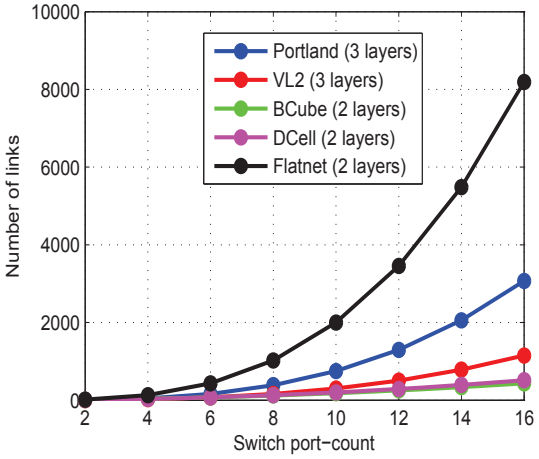


FIGURE 3.1: Comparison of number of links among various topologies.

3.2.1.2 Availability of 60 GHz wireless technology

The availability of 60 GHz technology strengthens our proposal to introduce it in Flatnet data center network.

TABLE 3.2: Ethernet vs 60 GHz links.

Feature	Ethernet	60 GHz links
Link speed	2-10 Gbps	6 Gbps
Bit error rate	10^{-13}	10^{-12}
Communication range	400-2000 m	10 m

Table 3.2 shows that wireless parameters are as good as Ethernet links which enables

60 GHz technology to be implemented in data center networks. The short communication range of this technology will not be a problem in data center environment because the equipment will be close to each others. Moreover, 60 GHz beamforming radios are affordable and available either as antenna arrays or horn antennas[39].

3.2.2 wFlatnet network structure

3.2.2.1 Physical structure

As described previously in chapter 2, Flatnet comprises two layers. The first layer consists of a switch connected to n servers. The second layer includes n^2 1-layer Flatnets linked using n^2 switches. We aim at creating shortcuts to connect servers and avoid long paths. Adding wireless links to every server can result in a huge number of radios and will lead to a waste of resources and the increase of the network cost. So, it is more convenient to add a radio to a group of servers. Since Flatnet is composed of connected subsystems (1-layer Flatnets), we propose to add a radio to every component of the system. Thus, every switch that groups n servers will be equipped with a wireless link. Servers can communicate with other groups passing by their switch which will use its antenna to reach destinations. Cables defined in the original topology will be maintained as illustrated in Figure 3.2.

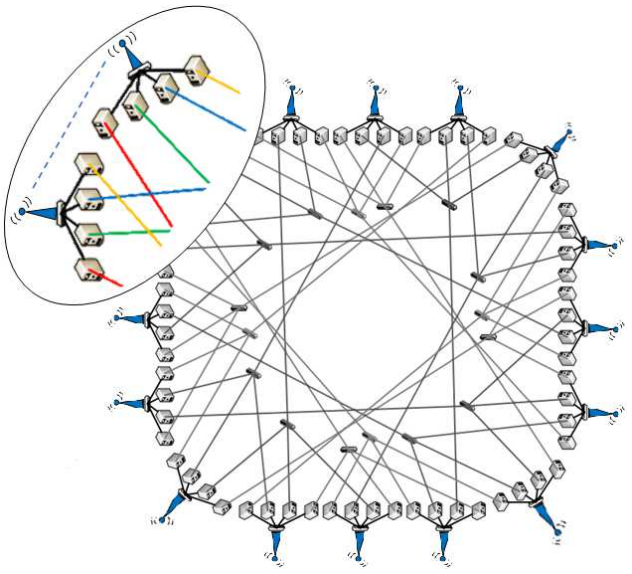


FIGURE 3.2: wFlatnet.

To minimize the power consumption, increase the signal directivity and concentrate the transmission energy to avoid multipath effect, 60 GHz beamforming technique is adopted [67]. However, this technique requires Line Of Sight (LOS) between the sender

and the targeted receiver. So, connected equipment are limited by the neighboring radios and reaching distant servers needs multi hops [68]. In addition, directional links can produce interference with receivers near the targeted device which consequently limits the number of concurrent transmissions. But, the main goal is to enhance the current architecture so that every 1-layer switch can be connected to all other 1-layer switches. Therefore, the requirement of LOS must be surpassed. A non line of sight method to address these issues was proposed in [38][39] and called 3D beamforming (see section 2.5.2). This technique uses ceiling reflections to connect non neighboring racks. Indeed, each sender positioned in the top of the rack will point its beam to the ceiling that will reflects it to the desired receiver as shown in Figure 2.16 in chapter 2. Thereby, beamforming provides an indirect reliable communication where blocking obstacles are avoided, the reach of radio signals is extended and pairs of switches are connected by only one hop.
As described in the previous chapter, 3D beamforming needs 3 components: Beamforming Radios, ceiling reflectors and electromagnetic absorbers. In practice, common data centers are composed of racks grouped in 5x5 clusters and every cluster contains 10 racks. Each rack can host up to 80 networking equipment. Moreover, standards have fixed the size of racks to be 4ft x 2ft[37] and since the radio antenna is sized 1ft x 1ft, we can place up to 8 radios per rack.

TABLE 3.3: Number of racks to host wFlatnet.

Switch port-count (n)	No. 1-layer wireless switches	No. servers + 2-layer switches	No. needed racks
4	16	80	2
8	64	567	8
16	256	4352	58
20	400	8400	110
24	576	14400	188
26	676	18252	237

Because of these physical constraints, we will place only 8 1-layer switches among equipment in every rack and fix their antennas on top. Table 3.3 demonstrates that a common 250-racks data center can host a wFlatnet where switch port-count is up to 26 and top of racks suffices for all the antennas. A larger data center can host a bigger wFlatnet without any physical restriction.

3.2.2.2 Radio Interference

In order to maximize the number of concurrent links and minimize the interference, we based our work on the prototype done in [39] with the same configuration of wireless links. Under the configuration of [39] (i.e. 8 radios in top of each rack, 3 channels of 60GH radios, and a ceiling of 3m) wFlatnet would have 390 bi-directional links simultaneously. It means 780 1-layer switches can communicate concurrently which fulfills our proposal of wFlatnet data center when n is up to 26.

3.2.2.3 Wireless adapted shortest path routing scheme

The wFlatnet intends to replace all 6 and 8 routing paths by shorter routes using wireless shortcuts. So, the algorithm that guarantees the best performance in term of path length and latency is the shortest path routing scheme. For our case, in order to compute the shortest path, wFlatnet is divided into subsystems. Every 1-layer wFlatnet will present a subsystem. Two coordinates (C_2,C_1) will label each server that corresponds to the server number C_1 of the C_2^{th} subsystem. Let's assume that a server (S_2,S_1) has to communicate with a server (D_2,D_1). Depending on the number of hops in the wired route, the wireless adapted shortest path scheme will be executed. Thus, the maximum path length will not exceed 4 hops. Algorithm 1 presents 3 routing cases in a fault-free environment. In fact, if the source and the destination are in the same subsystem, a wired route of 2 hops will be exhibited. If they are in different subsystems and the number of wired route hops exceeds 4 hops, the (S_2,S_1) will send the transmitted packet to its S_2 1-layer switch which will forward it wirelessly to the D_2 switch to reach finally the server (D_2,D_1). This path will take only 3 hops. Concerning the wired paths, Flatnet routing algorithm is adopted.
The wireless adapted routing scheme is accompanied with a fault-tolerance algorithm to satisfy the reliability and load-balancing needs of the data center network. Indeed, routing tables can be changed in real time in case of failure. Fault-tolerance scheme described in Algorithm 2 adds two supplement cases. First, when the wired path fails and if there is no other path shorter or equal to four, the wireless link is adopted. Second, if the wireless path fails because of technical issues or a probable interference, the wired path is available.

3.2.3 System evaluation

This section presents the simulations conducted in order to evaluate the performance of the proposed wFlatnet architecture compared to Flatnet architecture. Various metrics

Algorithm 1 Wireless adapted shortest path routing algorithm.

Case-1 (2 hops)
if $S_2 = D_2$ **and** $S_1 \neq D_1$ **then**
  (S_2,S_1) **1-layer route**→(D_2,D_1)
end if
if $S_2 \neq D_2$ **then**
  Calculate Wired path
  if $num_{hops} \leq 4$ **then**
    Case-2 (≤ 4 hops)
    (S_2,S_1) **2-layer route**→(D_2,D_1)
  else
    Case-3 (3 hops)
    (S_2,S_1) **1-layer route**→(SW_{S2})
    (SW_{S2}) **Wireless Link**→(SW_{D2})
    (SW_{D2}) **1-layer route**→(D_2,D_1)
  end if
end if

Algorithm 2 Faulty-tolerant routing algorithm.

Case-4 (≤ 4 hops)
if Wired Path Fail **then**
  Calculate new Wired path
  if $num_{hops} \leq 4$ **then**
    Use new Wired path
  else
    (S_2,S_1) **1-layer route**→(SW_{S2})
    (SW_{S2}) **Wireless Link**→(SW_{D2})
    (SW_{D2}) **1-layer route**→(D_2,D_1)
  end if
end if
Case-5 (≤ 8 hops)
if Wireless Path Fail **then**
  Use Wired path
end if

are considered such as diameter, average path length, latency, power consumption, etc. Besides, the system is tested under various network conditions to better illustrate a realistic environment.

3.2.3.1 Average path length and Diameter

The average path length (APL) is an important parameter that determines the performance of the network since it has a great impact on the average delay of transmission. We applied a uniform traffic distribution with an All-to-All pattern to calculate the APL. Figure 3.3 shows the simulation results of n-sized wFlatnet and compares it to

Flatnet. It can be seen that the proposed solution provides a much shorter APL. In addition, the average path length is getting slightly shorter for bigger sizes of wFlatnet because the number of wired long paths grows and consequently processed with wireless shortcuts (3 hops).

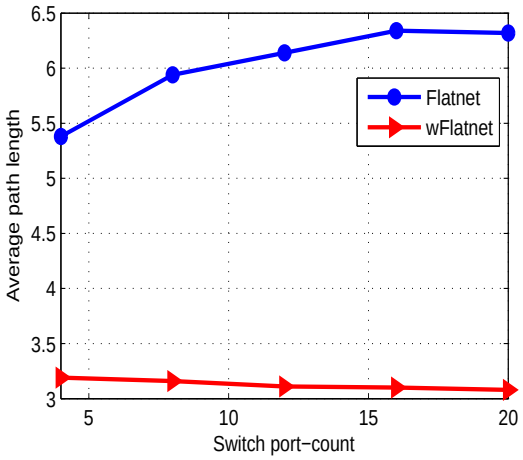


FIGURE 3.3: wFlatnet average path length compared to the Flatnet.

Also, we compared our proposed approach with other networks such as SprintNet[34]. In fact, SprintNet is a data center architecture that is designed to offer the shortest path length and diameter among all topologies (see chapter 2). However, wFlatnet outperforms SprintNet and DCell in terms of APL as shown in Table 3.4 and passes from a diameter equal to 8 in the original topology to a diameter equal to 4.

TABLE 3.4: Comparison of APL and Diameter among different topologies.

	512-server wFlatnet	600-server SprintNet	600-server DCell	512-server Flatnet
APL	3.16	3.77	4.72	5.94
Diameter	4	4	5	8

3.2.3.2 Network average latency

The network average latency indicates the average lifetime of packets to travel from the source to the destination. Latency depends on several parameters such as link types, switches processing, queuing delay, etc. We tried to simulate similar characteristics of data center architecture using ns-3 simulator[69] to create the Flatnet and the wFlatnet. Table 3.5 shows the simulation settings. We have used equipment with realistic IP addresses. The pattern of the traffic flow follows an exponential random distribution which is similar to the realistic data centers [70]. Figure 3.4 demonstrates clearly the

contribution of the wFlatnet to achieve a better performance compared to Flatnet. In addition, a larger network achieves larger latency reduction.

TABLE 3.5: Simulation settings.

Packet size	1024 bytes
Data rate of packet sending	1 Mbps
Data rate for device channel	3000 Mbps
Communication pairs selection	Random selection with uniform probability
Traffic flow pattern	Exponential random traffic

Table 3.6 shows transmission delays of different routing cases and details how the proposed wireless solution reduced the average latency by more than the half.

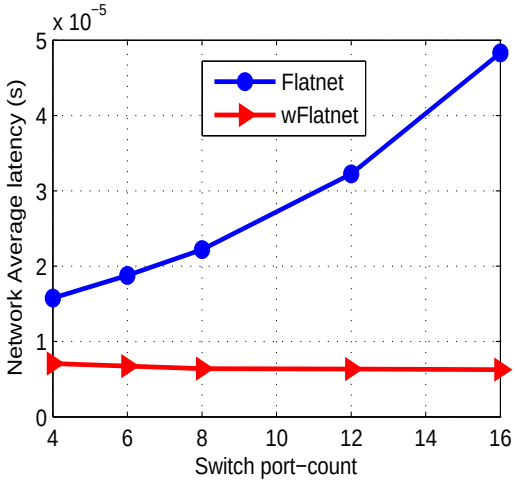


FIGURE 3.4: Average network latency.

TABLE 3.6: Average transmission delay.

Routing cases	Flatnet		wFlatnet	
	No. hops	Average latency (s)	No. hops	Average latency (s)
Case 1	2	5.65 $10^{-}6$	2	5.65 $10^{-}6$
Case 2	max 4	1.10 $10^{-}5$	max 4	1.10 $10^{-}5$
Case 3	6 8	1.70 $10^{-}5$ 3 $10^{-}5$	3	7.9 $10^{-}6$

3.2.3.3 Aggregate bottleneck throughput

The ABT is used to estimate the network capacity under All-to-All traffic pattern (see section 2.4.5 in chapter 2). After calculating the ABT of a 4096 server network (16-sized

topology, n =16), the wFlatnet has proved its capacity of improving the ABT by reaching 5367 instead of 2095 accomplished by Flatnet topology. This result can be explained by the short average path length (APL) that impacts considerably the throughput of the network.

3.2.3.4 Performance under faulty conditions

Three types of failures can affect a data center. The first one is the link failure because of cable or card malfunction and leads to connection failure between devices. The second one is the node failure that occurs when there is a hardware problem in a server/switch. The third failure is the mis-wiring that appears when two nodes are not correctly linked and this link becomes useless since it is not introduced in the routing table. When the server is connected to more than one link, it can be reached from other routing paths, which is the case in server centric architectures.
Flatnet is tested under this realistic environment where failures can occur. Figure 3.5 presents the performance of 1728-servers wFlatnet (12-sized topology, n=12) compared to the same sized Flatnet. In fact, the integration of wireless antennas in all 1-layer switches gives Flatnet a robust tolerance to faulty conditions. We have applied an All-to-All traffic pattern to the two topologies and randomly failed some switches. Since all servers can communicate using wireless shortcuts and passing only by the 1-layer switches, the failure of all 2-layer switches does not impact the performance of the wFlatnet as shown in Figure 3.5(a) unlike Flatnet topology which depends on the second layer to communicate between subsystems. Figure 3.5(b) and 3.5(c) show the resilience of the wireless architecture to the random failures of both first and second layer switches. Actually, this solution presents a diversity of paths since in case of any failure of antennas or unavailability of wireless transmission channels, a wired 2-layer route can be proceeded. For the failure of wired routes, communicating using wireless paths is a solution which not only grants more backup routes but can also minimize the latency as seen in Figure 3.5(c) where the wFlatnet offers a low average packet lifetime. In addition, Flatnet is defined as a server centric architecture where servers contribute in the routing procedure. So, servers failure will greatly impact the performance of the network. However, our new routing algorithm minimizes the routes passing by servers to 28.5% of All-to-All routes when $n = 4$, 19.7% when $n = 8$ and 10.9% when $n = 16$. Thus, there is no need to consider the case of servers failure since it will not have a big impact on the performance of the wFlatnet.

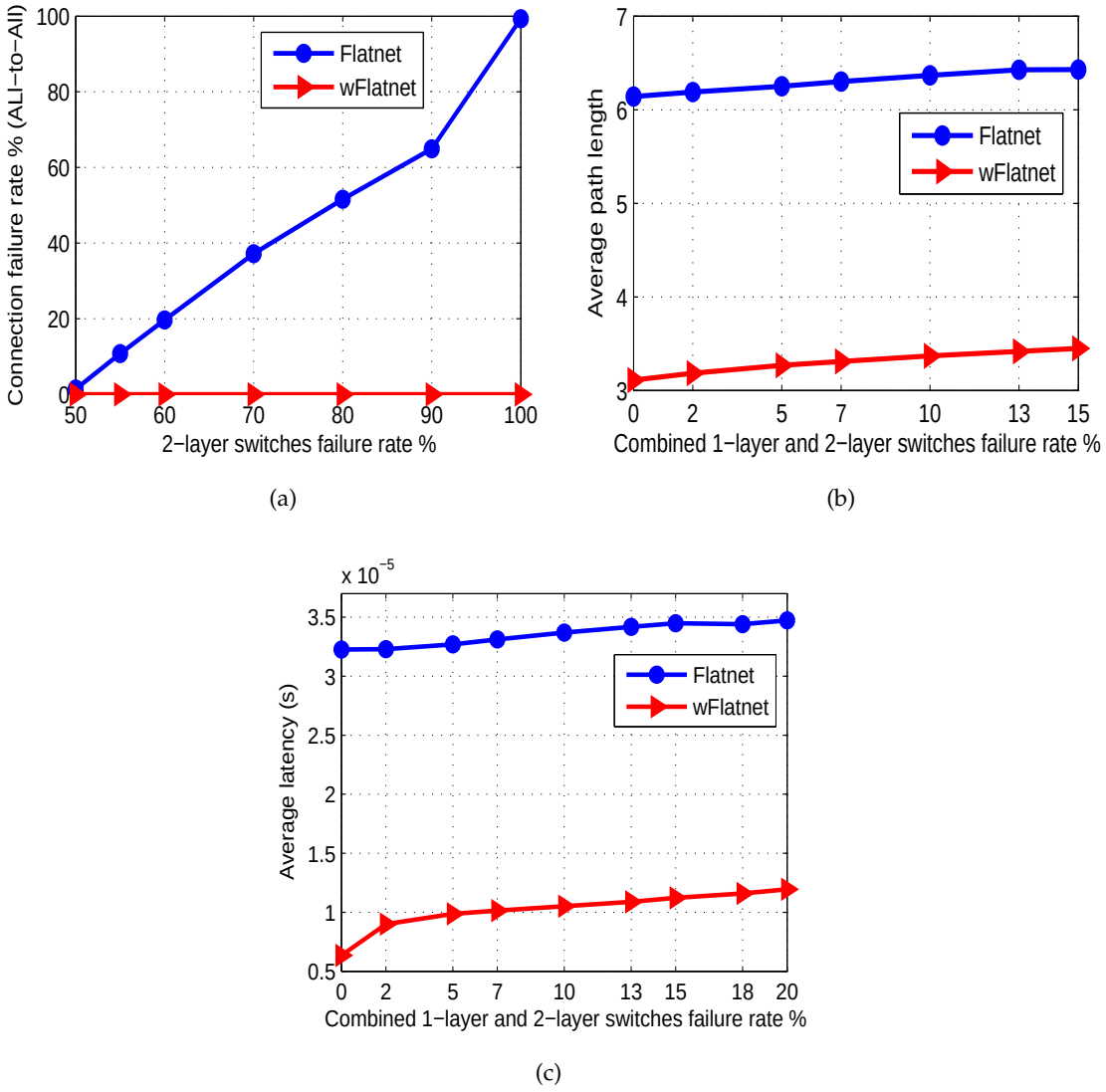


FIGURE 3.5: Performance of a 1728-servers wFlatnet under faulty conditions compared to the Flatnet.

3.2.3.5 Cost

Besides augmenting the performance of the Flatnet, our proposal comes with a simple solution that is easy to implement and maintain. In addition, compared to switches and links, 60 GHz transceivers are excepted to be inexpensive [2] and will not present a burden on the price of the network. So, using wireless technologies is a cost effective solution unlike approaches proposing adding more equipment to solve data center issues. Furthermore, reflectors used in the ceiling to reflect the signal and connect non-neighboring equipment do not require specialized metal and can be implemented using a low cost reflecting plates [39].

3.2.3.6 Power consumption

Data centers are one of the biggest and fastest growing consumers of electricity. Therefore, power consumption has become one of the important performance criteria. So we tried to estimate the utilization of equipment for a one-to-all traffic pattern of 4096-servers network (16-sized Flatnet, n=16). In fact, when a server communicates with all other servers, we calculated how many times different switches or servers have processed the data and how many times different transmissions have used links. For example, a server can communicate with $2(n-1)$ destinations through only 2 hops. For 2 hops path, the communication flow needs to pass by two links and one switch. So, to communicate with all 2 hops-destinations, $2(n-1)$ operations are processed by switches and $4(n-1)$ transmissions by links are accomplished. Figure 3.6 shows the difference between Flatnet and wFlatnet in terms of number of operations completed by switches and servers as well as the number of link transmissions. Networking equipment in wFlatnet accomplish the transmission tasks before Flatnet and switch to idle state. Knowing that idle equipment can save up to 20% of energy[71], we can consider the wFlatnet as a power effective solution. Regarding the transceivers, their energy consumption is minor. Indeed, the maximum power consumption of 60 GHz transceiver is 0.3 watts[1] which is minimal compared to 12 Watts, the maximum power consumption of a switch port.

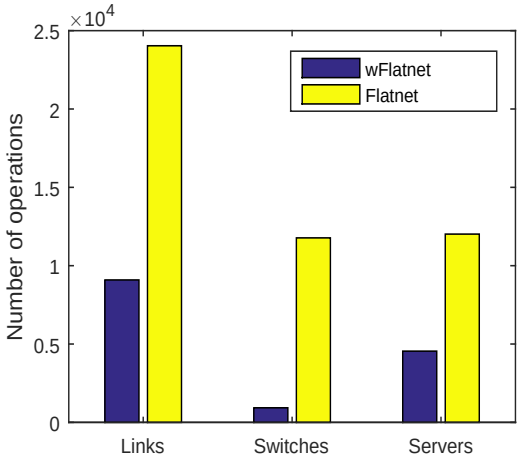


FIGURE 3.6: Number of operations completed by switches, servers and links.

WFlatnet is designed for the time-sensitive data centers since it enjoys a reduced average latency and offers a rapid services to the final users. However, wFlatnet is destinated for small data center networks. In fact, the 3D beamforming technology needs to be installed in one room so that the switches can communicate. Hence, the size of the DC is limited by the size of the room which can not host thousands of network

equipment. Based on this issue, a new data center interconnection is proposed to make a trade-off between scalability and latency of the network.

3.3 Improving the scalability: Parametrizable topology (PTNet)

This section presents PTNet[72], a new data center topology that is specifically designed to offer a high and parameterized scalability with just one layer architecture. In fact, PTNet is based on cells to build its topology. The size of each cell can be adjusted by a parameter s to make PTNet a parameterisable architecture that constructs a high scalable or small network according to the operators requirements. Furthermore, despite its high scalability, PTNet grants a reduced latency and a high performance in terms of capacity and fault tolerance. Consequently, compared to widely known data center networks, our new topology shows better capacity, robustness, cost-effectiveness and less power consumption. Theoretical analyses and conducted experiments illustrate the performance of the novel system.

3.3.1 Motivation

3.3.1.1 Trade-off between scalability, path length and cost

The network scalability leads to the usage of more routing equipment, more wiring complexity, more power consumption, higher cost and longer path length. For example, DCell is a data center enjoying a double exponential scalability with a robust routing algorithm. However, it does not support short paths, as routes in a large DCell network can reach more than 40 hops (see section 2.6.1.1 in chapter 2). Another example of a well-known data center is BCube. Unlike DCell, BCube is highlighted by its small number of hops between servers. However, BCube topology fails to scale to a larger network. Thus, more efforts should be carried out to design a data center network with a trade-off between the aforementioned facts.
In addition, to scale up, DCell and BCube architectures require at least three layers. Considering the fact that designing a layered data center is expensive in terms of cost and complexity, recent researches have opted for the design of a flat architecture, which presents several advantages:

- Simplifying the architecture: servers are connected without adding extra devices to scale to a larger network. In fact, flat networks are designed to connect the small segments of the network using one device instead of adding several switches or ports of servers to connect higher levels in hierarchical network designs.

- Segmenting the network into simple blocks: machines in a flatter architecture can be segmented into simple and small partitions with higher performance and more flexibility unlike recursive architectures segmented into several layers.
- Reliability: a simpler architecture can reduce routing complexity and minimise operation errors and failures. Indeed, designing a routing algorithm based on small parts of the network composed of smaller number of devices is simpler than conceiving routes between layers composed itself from different clusters.
- Power saving: a flat architecture implements less equipment which requires less power consumption.

There have been several initiatives that aim to design flat architectures including Juniper networks[73] and Cisco ACI Fabric[74]. Flatnet data center[33] is also a proposal of a flatter topology that scales at a high speed, but it suffers from a high delivery delay. This influenced the design of the proposed architecture, namely PTNet, that combines the benefits of one-layer data center, scaling with a complexity of sn^2 (where n is the number of ports in a PTNet switch and s is a parameter to vary the range of scalability of the network) as well as offering an efficient routing between servers (minimum path length).
For end users, efficiency in service execution is crucial. So, the packet delay is an important requirement that needs to be taken into account. A solution could be to connect servers directly without passing by many intermediary switches. In a few server centric architectures, such as BCube and Flatnet, servers have already forward units and two built-in NICs[75] dedicated to forward packets to an additional switch. This induces an additional packet processing and queuing delay. Firstly, getting rid of some intermediary switches can minimise the path between two nodes; and secondly it decreases the complexity of cabling, installation and maintenance.

3.3.1.2 Gradual scalability

Gradual scalability is also a critical aspect for a data center administrator. When he/she needs a specific number of servers to build a network, he/she should not be obliged to implement a much bigger number of servers than needed. Let us consider Figure 3.7 that shows the scalability of DCell and Flatnet networks. This Figure shows that DCell scales double exponentially, meaning that when the number of ports per switch is increased by 2, the number of servers increases to around the double. If DCell network is to be implemented with 10 000 servers, then 10-ports switches will need to be deployed. There will be then 2210 extra servers (12210-servers network) in addition to the extra

switches and links. Obviously implementing extra number of devices will consume more energy. Flatnet suffers from similar scalability problem but with a lower degree compared to DCell. Thus, the design of a parameterisable architecture, where the range of size of the network can be specified, is needed with a transition from a small system to a large one being gradual.

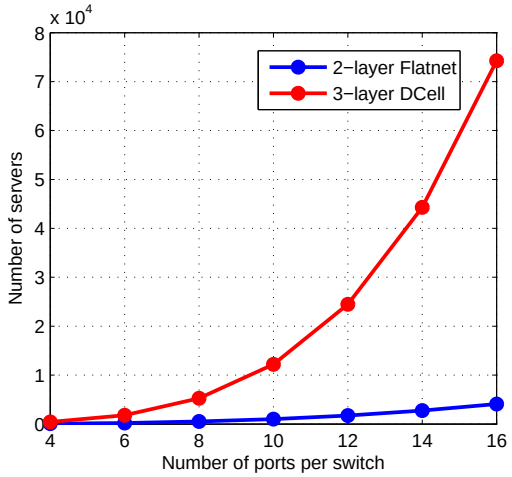


FIGURE 3.7: Scalability of DCell and Flatnet.

3.3.2 PTNet network structure

PTNet has four properties that address the aforementioned challenges: a scalable and parameterisable structure, an efficient routing algorithm that guarantees a minimum path length, fault-tolerant routing that addresses links, servers and switches failures and congestion situations and an incremental upgrade structure that offers a gradual expansion of the data center size.

3.3.2.1 Physical structure

PTNet uses n-ports switches and multiple-ports servers (two ports and more). Some servers are connected to several other servers via bidirectional links, as it will be explained later. PTNet is composed of n cells. Each cell contains s switches. Every switch is connected to n servers. Thus, PTNet can house sn^2 servers and sn switches. Cells in PTNet are labeled $i = 1..n$. Switches in cell i are labeled (i,j) where j= $1 \cdots s$. And, servers connected to a switch j in a cell i will have the coordinate (i,j,k), where

k=1$\cdots n$. Let switches that have the same coordinate j in different cells be called *homologue switches*. For example, switches (2,1) and (3,1) are homologue switches. Furthermore, a server with coordinate (i,j,i), which means k=i, is called *master server*. As an example, all servers (1,j,1) are the master servers of the first cell.

Algorithm 3 summarizes the different steps to connect PTNet servers. Two steps are proposed to build PTNet: If cells contain more than one switch ($s > 1$), an intra-cells connection is created: within a cell, all masters must be connected. The second step is to connect inter-cells: a master with coordinate (i,j,i) will be connected to the i-th servers of the j-th switches of all other cells. Figure 3.8 shows an inter-cells connection of a 16-servers PTNet, where n=4 and s=1, and Figure 3.9 depicts an intra-cells connection of a PTNet with s=4 and n=4 (one cell is presented).

Algorithm 3 BuildPTNet

Input: n (Number of ports per switch), s (Number of switches per cell), node (List of nodes)
Output: node (Updated list of nodes)
for $i = 1..n$ **do**
  for $j = 1..s$ **do**
    if $s > 1$ **then**
      Connect intra-Cells servers
      for $k = j + 1..s$ **do**
        Connect (node (i, j, i), node (i, k, i))
      end for
    end if
    Connect inter-Cells servers
    for $k = i + 1..n$ **do**
      Connect (node (i, j, i), node (k, j, i))
    end for
  end for
end for

In summary, using the proposed algorithm, every switch in the network has a server with coordinate (i,j,i), called master server, which will be connected to all masters in the same cell and the i-th servers of the homologue switches in other cells.
Table 3.7 describes some features and structural proprieties of PTNet as well as showing a comparison with other data center networks. These properties are calculated as follows:

- The number of links is equal to $sn(2n-1) + n\sum_{i=1}^{s-1} i = sn^2 + sn(n-1) + n\sum_{i=1}^{s-1} i$:
  - We have sn^2 switch-to-server links (all servers are connected to a switch).
  - We have $sn(n-1)$ inter-cells connections (we have sn master servers and each master is connected to $(n-1)$ clusters).

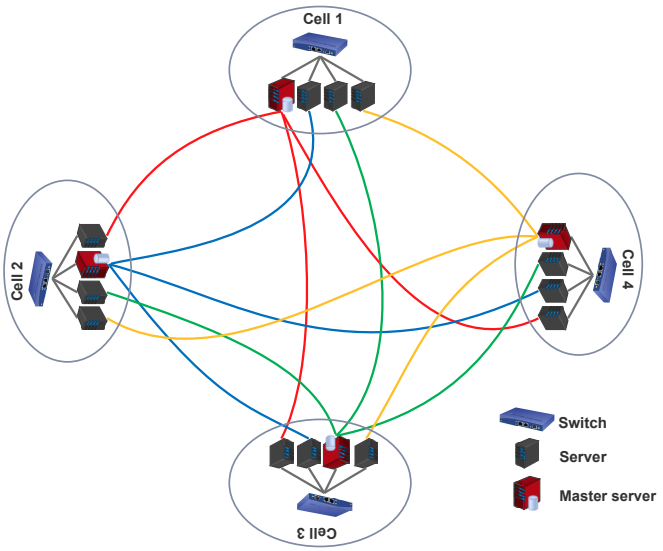


FIGURE 3.8: An inter-cells connection of a 16-servers PTNet using one 4-port switch in each Cell.

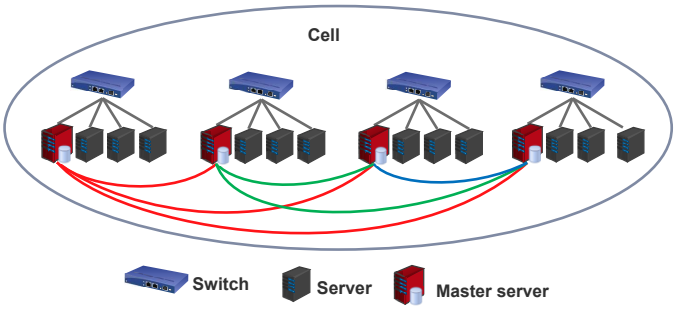


FIGURE 3.9: An intra-cell connection of a PTNet where s=4 and n=4.

  - We have $n \sum_{i=1}^{s-1} i$ master-to-master/intra-cell links (The first master connects the $(s-1)$ following masters, the second connects the $(s-2)$ following masters,..., the final master will be automatically connected by the previous ones). This connection is done for all the n clusters.

- The number of links per server can be equal to 2 for a normal server and more than 2 for a master server.
- Switches per server $= \dfrac{\text{Number of switches}}{\text{Number of servers}} = \dfrac{sn}{sn^2} = \dfrac{1}{n}$
- In PTNet data center network, we have n clusters. In each cluster, we have s master servers. Hence, after cutting the network into 2 equal halves, we will have $\frac{n}{2}$ cluster and $\frac{sn}{2}$ master servers in each half. Each master is connected to a non-master server from each cluster from the other half. So, the number of links connected to the other half is equal to $(\frac{n}{2} * \frac{sn}{2})$. Since we have 2 halves, and each

half connects $\frac{sn^2}{4}$ servers from the other half, the number of wires linking the two halves is equal to $\frac{sn^2}{2}$. Therefore, the bisection width is equal to $\frac{sn^2}{2}$.

- Bisection width per server $= \frac{\text{Bisection width}}{\text{Number of servers}} = \frac{sn^2}{2sn^2} = \frac{1}{2}$
- The diameter is calculated by simulation. In fact, by building the graph of the topology and calculating the length of all shortest paths between all nodes, we can find the maximum shortest path.

TABLE 3.7: Properties of different network architectures.

	DCell (2 layers)	BCube (2 layers)	Flatnet (2 layers)	PTNet
servers number	$n(n+1)$	n^2	n^3	sn^2
Links number	$\frac{3n(n+2)}{2}$	$2n^2$	$2n^3$	$sn(2n-1)$ $+n\sum_{i=1}^{s-1} i$
Per server	$\frac{3}{2}$	2	2	≥ 2
Switches number	$n+1$	$2n$	$2n^2$	sn
Per server	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$
Bisection width	$\frac{n^2}{4}+\frac{n}{2}$	$\frac{n^2}{2}$	$\frac{n^3}{4}$	$\frac{sn^2}{2}$
Per server	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$
Diameter	5	4	8	5

3.3.2.2 PTNet gradual scalability

Using PTNet topology, we can build a large data center network with a flat architecture. By deploying sn switches, where n is the number of ports per switch, PTNet hosts sn^2 servers which is approximately s times that of a 2-layer BCube and DCell. If $s = n$, PTNet has the same number of servers as Flatnet and can scale at a speed of n^3 which means if $s > n$, PTNet data center presents a larger network than Flatnet. In addition, the number of links in PTNet changes depending on the value of s, but still increases at the speed of $O(n^2)$. This degree of wiring is approximately the same as 2-layer DCell and BCube, and outperforms Flatnet. As discussed in the previous section, high scalability is not the only requirement of a data center network. Additionally, scalability must be gradual and the transition from a small network to larger one needs to be progressive. Specifically, a gradual expansion relates to the ability to build a specific size of a network without being constrained to implement more devices than needed. Unlike the scalability of DCell, which is depicted in Figure 3.7, our aim is to

design a parameterisable architecture where the administrator can control the desired range of size of a network. Indeed, PTNet size depends on the parameter s, which corresponds to the number of switches in a cell in addition to n the number of servers per switch. Therefore, we can obtain a small network when s has smaller values and reach a large network when s increases. Then, after adjusting the number of switches per cell, n needs to be updated to attain approximately the desired number of servers. Figure 3.10 shows different ranges of sizes that PTNet supports when varying s and increasing n. We can notice that this new topology scales well when augmenting s and offers multiple ranges of sizing starting from small to huge networks.

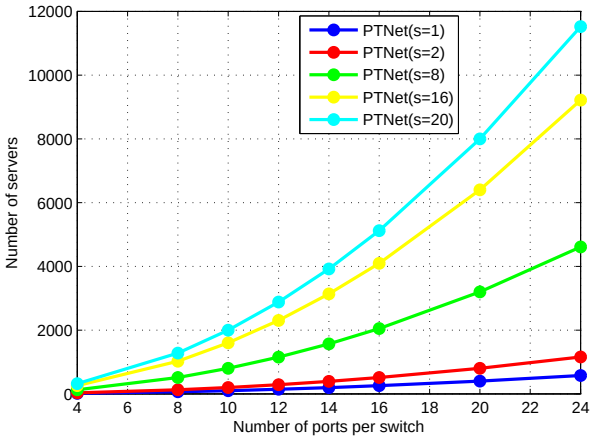


FIGURE 3.10: PTNet size according to the number of switches per cell.

3.3.2.3 PTNet shortest path routing scheme

PTNet is characterised by its robust connection pattern and multiple paths between any two nodes. The connection pattern has an essential role to guarantee a resilient routing algorithm:

- More disjointed paths between two nodes will help the system to deal better with faulty conditions (see chapter 2, section 2.4.10).
- Shorter routes will help to enhance the efficiency of packet delivery.

The more number of disjointed paths a given network has, the more fault tolerant it becomes. In server centric topologies, a data center network has 2 node/edge disjointed paths between any two nodes because each sender/receiver has 2 ports to deliver or receive a packet. In PTNet, between any two master servers we have $(n + s - 1)$ routes. This increases the average number of disjointed paths compared to typical server centric data centers (DCell, BCube, Flatnet, etc.) and consequently increases the robustness

of the system.
PTNet routing algorithm presented in algorithm 4 is a customized one that is based on shortest path. It is designed to be applied to the network where all servers are included in the routing process. It computes the routing tables once, and then routing paths are stored and retrieved every time a packet needs to be sent. To send a packet from a node S (S_3,S_2,S_1) to a node D (D_3,D_2,D_1), an inter or intra cell route is calculated, according to the position of S and D. The computed path does not exceed five hops, and if the source S and the destination D are located in the same cell, the routing procedure can be divided into three scenarios:

- If S and D are connected to the same switch, they can directly reach each other via this switch.
- If S and D are connected to different switches and $S_1 \neq D_1$, the traffic must pass by the master servers then it is routed to the destination.
- When $S_1 = D_1$ and the source and the destination are in different switches, an inter-cell routing is proceeded and the traffic passes by an intermediate homologue switch connected to the destination.

If S and D are in different cells: whether connected to homologue switches or not, the traffic must pass by an intermediate homologue switch connected to the destination.
Table 3.8 shows the path distribution of 1200-servers-PTNet ($n = 20$, $s = 3$). We can see that the paths length does not exceed 5 and routes lengths $\in$ [1,2,3,4,5].

TABLE 3.8: Flows distribution in all-to-all communication for 1200-servers PTNet (n=20, s=3).

All-to-all traffic pattern & uniform distribution flow mode	
Path distribution (1438800 routes in total)	2400 paths of length 1
	47880 paths of length 2
	132240 paths of length 3
	517560 paths of length 4
	738720 paths of length 5

In addition to the shortest path routing scheme, a fault tolerant routing algorithm is proposed to protect the network from failures as well as to find alternative path. A failure can occur in case of malfunction of a server, a switch or a link or when the bandwidth is not available because of a congestion. In this case, the new routing decisions

(described in Algorithm 5) resume the previously failed communication and guarantee that the packets will be delivered as fast as possible to their destinations. The advantage of this algorithm is that it suggests alternative short routing paths, as PTNet provides various routes in its fully connected topology. In fact, any server can reach its destination passing by any cell from at least n paths. These paths, that may share some nodes or links, contribute to achieve a good load balancing and fault tolerance. The routing path with the most available bandwidth is chosen.

Algorithm 4 Fault Free Routing Algorithm

Input: (S_3, S_2, S_1) (Coordinates of source server),
(D_3, D_2, D_1) (Coordinates of destination server), (x, y) (Coordinates of a switch), $x, y \in [S_3, S_2, S_1, D_3, D_2, D_1]$
Output: Path
if $S_3 = D_3$ **then**
  if $S_2 = D_2$ **then**
    Case 1 (2 hops)
    Source and destination are in the same Cell and connected to the same switch.
    Path= $(S_3, S_2, S_1) \rightarrow (S_3, S_2) \rightarrow (D_3, D_2, D_1)$
  else
    if $S_1 = D_1$ **then**
      Case 2 (3 hops)
      Source and destination are in the same Cell and connected to different switches in the same position.
      Path= $(S_3, S_2, S_1) \rightarrow (S_1, S_2)$ master server $\rightarrow (S_1, D_2)$ master server $\rightarrow (D_3, D_2, D_1)$
    else
      Case 3 ($\leq$ 5 hops)
      Source and destination are in the same Cell and connected to different switches in different positions.
      Path= $(S_3, S_2, S_1) \rightarrow (S_3, S_2) \rightarrow (S_3, S_2)$ master server $\rightarrow (S_3, D_2)$ master server $\rightarrow (D_3, D_2) \rightarrow (D_3, D_2, D_1)$
    end if
  end if
else
  if $S_2 = D_2$ **then**
    Case 4 ($\leq$ 4 hops)
    Source and destination are not in the same Cell but connected to homologue switches.
    Path= $(S_3, S_2, S_1) \rightarrow (S_1, S_2)$ master server $\rightarrow (D_3, S_2, S_1) \rightarrow (D_3, S_2) \rightarrow (D_3, D_2, D_1)$
  else
    Case 5 ($\leq$ 5 hops)
    Source and destination are not in the same Cell and connected to non-homologue switches.
    Path= $(S_3, S_2, S_1) \rightarrow (S_1, S_2)$ master server $\rightarrow (S_1, D_2)$ master server $\rightarrow (D_3, D_2, S_1) \rightarrow (D_3, D_2) \rightarrow (D_3, D_2, D_1)$
  end if
end if

Algorithm 5 Fault Tolerant Routing Algorithm

Input: (S_3, S_2, S_1) (Coordinates of source server), (x, y) (Coordinates of a switch), $x, y \in [S_3, S_2, S_1]$, n (Number of ports per switch)
Output: Path
if case 1 fails **then**
  Path= $(S_3, S_2, S_1) \rightarrow (S_1, S_2)$ master server $\rightarrow (S_i, S_2, S_1) \rightarrow$ use Fault-Free algorithm
  $S_i \in [1..n] \backslash \{S_1, S_3\}$ chosen depending on bandwidth
end if
if cases 2,3,4 or 5 fail **then**
  Path= $(S_3, S_2, S_1) \rightarrow (S_3, S_2) \rightarrow (S_i, S_2, S_1) \rightarrow$ use Fault-Free algorithm
  $S_i \in [1..n] \backslash \{S_1\}$ chosen depending on bandwidth
end if

3.3.3 System evaluation

Several experiments were carried out to evaluate PTNet. The following metrics are used: APL (average path length), diameter of the network, throughput, latency, and bisection width. PTNet is benchmarked against well-known data center networks including BCube, DCell and Flatnet. Moreover, PTNet is evaluated under different faulty conditions (links, servers and switches failures).

3.3.3.1 Average path length and Diameter

To calculate the APL, we have applied a uniform traffic distribution with all-to-all pattern (each server communicates with all other servers in the network). Figure 3.11 shows the experimental results of the average path length. The value of s is gradually changing to construct similar network sizes to other topologies: PTNet outperforms the 2-layers DCell, BCube and Flatnet by achieving the shortest average path length. Furthermore, PTNet's APL increases slightly when augmenting the size of the network.
Regarding the diameter, PTNet outperforms Flatnet topology by three hops as showed in Table 3.7. In addition, compared to a 2-layers DCell, PTNet achieves the same diameter and it is slightly bigger compared to a 2-layers BCube. However, DCell/BCube can not scale to a larger network until they are upgraded to a 3-layers topology where the diameter is equal to 10 for DCell and equal to 6 for BCube. Thus, for a scalable network, PTNet has a lower diameter.

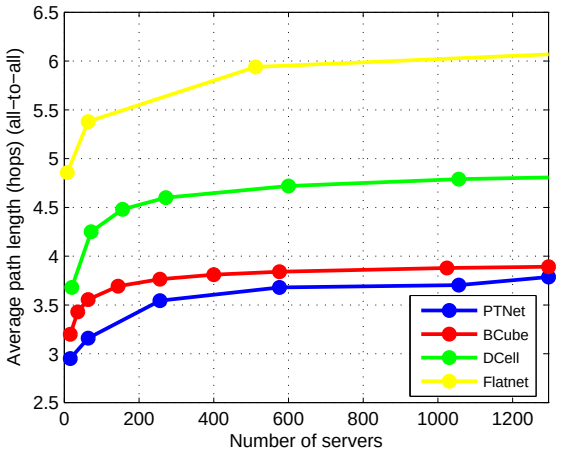


FIGURE 3.11: Average path length compared to different topologies.

3.3.3.2 Network average latency

As we described in chapter 2, the major contributor in the packet delay delivery is the waiting time in the queue and the time to process the data at each relay node. Therefore, offering several short paths to avoid congestion as well as connecting servers directly to pass by a minimum of switches can be the competitive edge of PTNet, unlike BCube and Flatnet, that use intermediary switches between any two servers.
The latency is computed for a some-to-some pattern to evaluate the network in a realistic environment. A traffic distribution is randomly created after fixing a percentage of nodes sending or receiving transmissions. The traffic consists of randomly selected pairs of nodes that exchange a 1Mbps flow of data. Realistic IP addresses with 1000 Mbps Ethernet switches are used. The simulation settings are summarized in Table 3.9.

TABLE 3.9: Simulation settings used for the development of different topologies.

Packet size	1024 bytes
Data rate of packet sending	1 Mbps
Data rate for device channel	1000 Mbps
Communication pairs selection	Random selection with uniform probability
Traffic flow pattern	Exponential random traffic

Figure 3.12 strengthens the theoretical analysis and shows that PTNet outperforms other topologies in terms of average transmission delay. Furthermore, when enlarging the network to more than 2500 servers, PTNet achieves around $\frac{1}{3}$ latency of DCell and

Flatnet. Also, PTNet latency increases gradually when the number of servers increases, which is not the case for DCell and Flatnet.

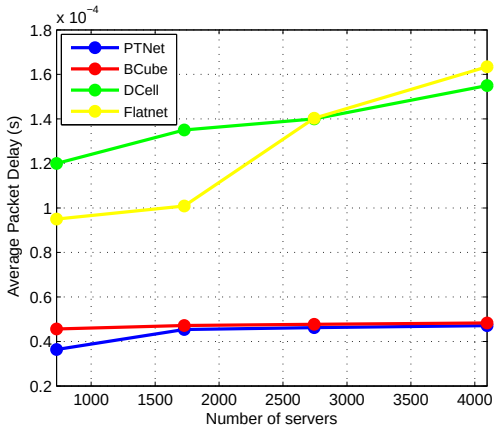


FIGURE 3.12: PTNet average latency compared to different topologies.

3.3.3.3 Aggregate bottleneck throughput

The aggregate bottleneck throughput is obtained by summing the throughput of all bottleneck flows. We assume that the bandwidth of all links in a one-way-communication is equal to 1 and that all links are two-way-communication (i.e. virtual links = 2 * physical links). We also assume that the overall capacity of the network (i.e. $C_{network}$) is N_{Vlinks}, meaning that it is equal to 2*N_{links}, where N_{Vlinks} denotes the number of virtual links and N_{links} denotes the number of physical links. if we assume that the proportion of the network capacity P_{ABT} can reach approximately $\frac{ABT}{C_{network}}$, we can prove that $P_{ABT} \approx \frac{1}{APL}$. A simple proof is as follows:

$$ABT \approx N_{flows} * \frac{1}{NF_{link}} \tag{3.1}$$

$$NF_{link} = \frac{N_{flows} * APL}{2 * N_{links}} \tag{3.2}$$

where N_{flows} is the number of flows in the network and NF_{link} denotes the number of flows carried in one link. We can conclude that:

$$ABT \approx \frac{2 * N_{links}}{APL} \tag{3.3}$$

$$P_{ABT} \approx \frac{ABT}{C_{network}} \approx \frac{1}{APL} \tag{3.4}$$

Equation 3.4 shows that the ABT is very sensitive to the APL: The ABT increases when the APL is the shortest. As seen in a previous section, PTNet owns the smallest path

length which gives this network topology the greatest performance in terms of aggregate bottleneck throughput.

3.3.3.4 Throughput

Based on the same simulation settings used to compare latency among different topologies, we have calculated also the average throughput of the network. The throughput of the network depends on the average packet delay (PTNet owns the smallest packet delay), the data rate of the channel (which is fixed to be equal for all topologies) and the rate of successful messages (PTNet routing algorithm offers multiple paths between servers).
Figure 3.13 shows that PTNet owns the best throughput, meaning that PTNet is both a robust and efficient network. PTNet shows a slightly better performance in terms of throughput compared to BCube because the average packet delay of both topologies is approximately similar. However, PTNet shows much better performance compared to DCell and Flatnet due to the difference in terms of latency and diversity of routing paths.

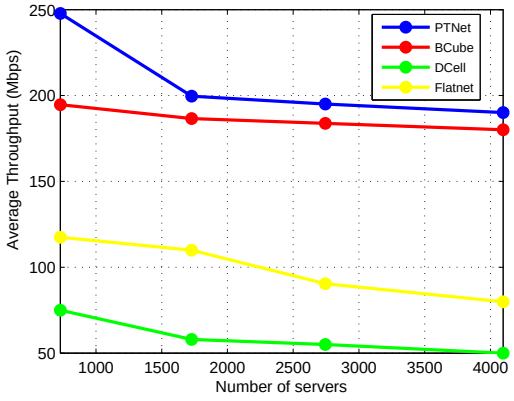


FIGURE 3.13: PTNet average throughput compared to different topologies.

3.3.3.5 Bisection width

The bisection width estimates the worst failure case of the system (see section 2.4.6 in chapter 2). The larger the bisection width is, the better fault tolerance the network will have. The bisection width of PTNet is $\frac{sn^2}{2}$ which is s times that of BCube and around $2s$ times that of DCell. As we can see from Figure 3.14, the bisection width increases when s increases. When $s = n$, it is two times that of Flatnet. Also, as shown in Table 3.7, PTNet outperforms the other topologies in terms of bisection width per server, meaning that PTNet is a robust system against faulty conditions.

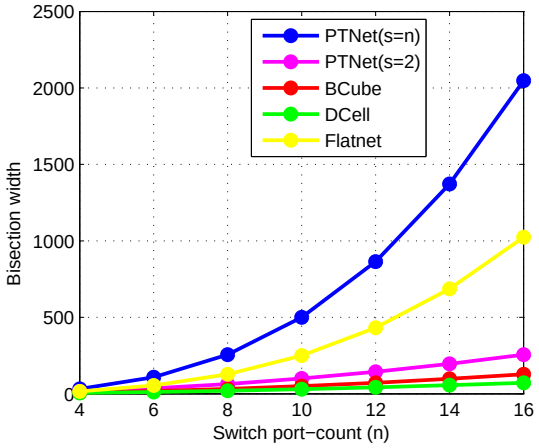


FIGURE 3.14: PTNet bisection width compared to different topologies.

3.3.3.6 Performance under faulty conditions

A device failure can have a impact on the network performance. Therefore, two elements of the network need to be checked: the fault tolerant algorithm and the connection pattern (that provides alternative routes with a minimum routing path length). PTNet proposes a trade-off between path length and fault-tolerance. Indeed, when the number of switches increases, which means s increases, fault-tolerance improves at the expense of the average path length (that slightly increases to reach maximum around 4.5). This is still acceptable compared to DCell and Flatnet.
Figure 3.15 shows the performance of 1200-servers PTNet $(s = 3, n = 20)$ under faulty conditions for all-to-all communication pattern. For links, servers and switches failures, APL increases slightly. For example, when the failure rate of servers, links and switches is 10%, APL reaches respectively 4.4987, 4.6173, 4.3519; this is an increase of 0.06% compared to the fault free case (APL = 4.35). Moreover, PTNet's APL under faulty conditions still offers better results than the average path length of Flatnet and DCell. Figure 3.15(c) shows that the diameter of the network increases in case of switches, servers and links failures and reaches respectively 6, 7, 9; this is an acceptable result compared to Flatnet topology that has a diameter of 8 in fault free case and more than 12 in faulty conditions[33]. Figure 3.15(b) shows that the ABT slightly decreases when failures occur. Finally, after failing randomly 10% of switches/links, experimental results demonstrate that all connections are maintained. When the servers failure rate is set to ρ, the connection failure ratio of the whole network does not exceed ρ. Knowing that the failure rate of networks devices cannot exceed 5% in real life data centers[76], PTNet is proved to have good performance and high robustness.

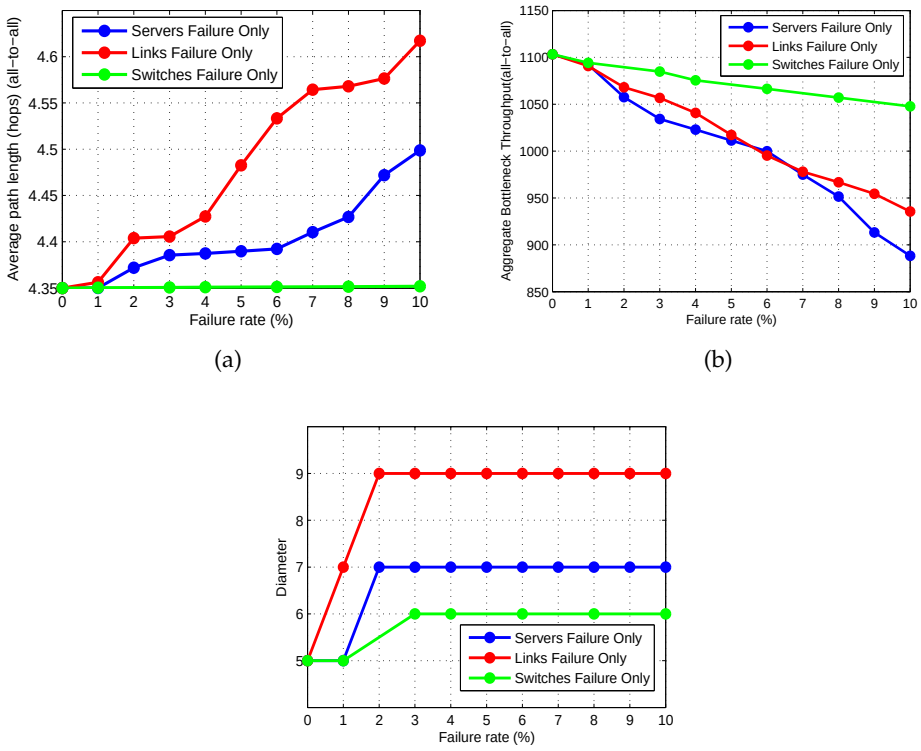


FIGURE 3.15: The performance of a 1200-servers PTNet under various faulty conditions (All-to-all communication pattern).

3.3.3.7 Cost

The continuous growing of data centers imposes an important criterion to be taken into consideration, which is the cost of the infrastructure. When enlarging the network, more cables and more equipment are used, meaning that the cost will increase. In terms of cost, an effective topology is the one that connects more servers with less equipment. Table 3.10 shows the number of wires and switches that need to be implemented to construct 2100-servers network. This data center can be built using a 2-layer DCell/BCube with a high-end switches or using a 3-layer DCell/BCube with a low-end switches. We also define equipment needed to build Flatnet as well as PTNet using high, mid and low-end switches by changing s. As we can see, with a high-end switches and 2-layer topology, BCube and DCell use respectively 96, 49 48-ports switches. However, with the same number of ports per switch, PTNet implements only 48 switches. By scaling to 3-layer topology, BCube and DCell use respectively 507 and 456 13-ports switches. In contrast, PTNet implements only 169 13-ports-switches which is less than BCube, DCell and Flatnet that implements 338 13-ports switches.

In addition to switches, PTNet uses multi-port servers that act like routers. To deliver packets to different cells, master servers have $(n + s - 1)$ ports. Adding NICs (more

ports) to a server can increase the cost of the network. However, this is a more economical than buying more machines (switches). Thus, we calculated the overall cost of the data center network. As representative of equipment prices, we use values of 450$ for 10Gbps Ethernet switch port, 150$ for 10Gbps port on an Ethernet NIC, and 200$ for server core[71]. Table 3.10 shows that PTNet owns a low cost despite adding multiple NICs to master servers. Regarding the wiring, PTNet owns an acceptable number of cables which is not superior to other topologies. To conclude, PTNet is considered as a scalable, low-path topology with a reasonable cost compared to other topologies.

TABLE 3.10: Cost comparison between different topologies.

Topology	number of servers	n	number of switches	number of wires	cost ($)
Flatnet (2 layers)	2197	13	338	4394	3075800
DCell (2 layers)	2352	48	49	3600	2234400
DCell (3 layers)	3192	7	456	5471	3514450
BCube (2 layers)	2304	48	96	4608	3225600
BCube (3 layers)	2197	13	507	6591	4394000
PTNet	2304	$n = 48$ & $s = 1$	48	4560	2520000
	2312	$n = 34$ & $s = 2$	68	4590	2522800
	2197	$n = 13$ & $s = 13$	169	4303	2366000

3.3.3.8 Power consumption

To estimate the power consumption of PTNet, we tried to estimate the CPU utilisation of a 48-sized network with an all-to-all traffic pattern and compare it to 48-sized DCell, 13-sized Flatnet and 48-sized BCube (approximately 2200 servers). Figure 3.16 shows how many times servers and switches processed the data (number of operations) for the aforementioned topologies. As PTNet has the smallest number of operations, devices will spend more time in idle state. Even though this state is inefficient, it consumes less energy than active state: idle equipment can save between 10 % and 20 % of energy [71][77]. We can conclude that PTNet is a power-aware data center. This is achieved thanks to its physical structure and its shortest path based algorithm that guarantee minimum operations done by the network and consequently minimum energy consumption.

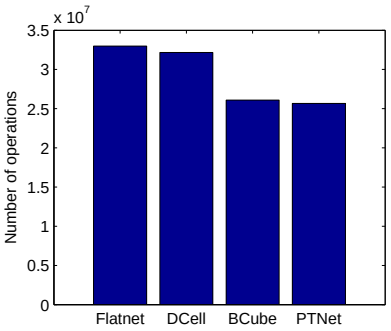


FIGURE 3.16: Number of operations among different topologies.

3.3.3.9 Performance of master servers

The master servers represent the competitive edge of PTNet topology. Thanks to the master servers, PTNet achieves the trade-off between the cost of the network, the performance and the energy saving. In fact, these servers play the role of relay nodes that forward packets between cells and minimise the number of intermediary switches. This contribution of master servers grants a better performance in terms of APL and latency. In addition, as explained in the previous section, NICs can be added to a server with a lower price than buying an additional switch (see Table 3.10). However, relying on master nodes can lead to an additional processing load compared to other servers. This eventual overhead is managed using the rich interconnection of the PTNet topology that offers multi routing paths and the fault tolerant routing algorithm 5, which selects the route with the most available bandwidth in case of congestion. Hence, lower cost and lower latency and APL are gained on the expense of a little concession in terms of performance due to the possible overhead.

3.4 Further discussion

Table 3.11 presents a summary description of the wFlatnet and PTNet. In fact, depending on the network operator requirements, we can choose the adequate design. If the installed applications are time sensitive, wFlatnet is a more suitable solution. In addition, for an existing network infrastructure that needs a performance enhancement, adding wireless links can be a rapid idea with a low cost and a simple installation. If the network needs to be scalable while offering a high performance, PTNet proved its efficiency compared to many existing solutions. In addition, for re-configurability (frequent resizing) requirement, PTNet offers an easy topology to parametrize.

TABLE 3.11: wFlatnet Vs PTNet.

Parameters	wFlatnet	PTNet
Scalability	Low	High
Latency	Very low	low
Fault-tolerance	Very high	High
Description	- A low cost/power consumption solution to enhance the average latency of Flatnet and design a network with a very low services execution time. - An efficient topology for time-sensitive data centers. - An easy installation for a performance enhancement solution.	- A low cost/power consumption solution to enhance the scalability of the network with a good average packet latency. - An efficient topology for a data center that needs a frequent re-configurability and resizing (parametrizable architecture).

3.5 Conclusion

In this chapter, we presented two designs for data center networks: The first topology is called wFlatnet, designed to enhance the average packet delay of Flatnet data center without burdening the network with additional cost and installation complexity. This novel design integrates wireless solution that acts as shortcut routes to minimize the delay of the network. The second proposed topology is called PTNet. This new network is highlighted by its parameterisable and gradual scalability while offering a low packet delivery delay. Another advantage of PTNet is that it relies on servers to forward packets and minimize the number of intermediary switches which helps at reducing the network overall cost. The implementation of these networks and the experimental results demonstrate their performances compared to well-known topologies and show the efficiency of their routing algorithms to enhance the latency and reduce the failures.

Chapter 4

Greening data center networks: Power aware routing algorithms

4.1 Introduction

Data centers are one of the largest consumers of energy in the world. In 2013, US data centers consumed a quantity of energy that can power the households of New York city for two years, which is equivalent to a one year output of 34 coal-fired power machines[78]. Hence, many concerns are raised about the huge amount of energy consumed by this gigantic infrastructure. Statistics in [79] and [80] show that the power consumed by all data centers in the world in 2011 accounts for 1.1%-1.5% from the world power consumption and it will increase to 8% by 2020. Therefore, appropriate solutions need to be developed to reduce the energy consumed by data centers.
The focus in this chapter is how to make the data center power efficient by designing two power aware routing algorithms characterized by a short computation time. The first algorithm is designed for PTNet topology described in chapter 3 while the second can be applied to any data center network. The idea is to switch on only ports of devices that send and receive packets and the ones that maintain the system non-partitioned. Then, we power off the devices that are not contributing in the packets transmission. The main contributions of this work can be summarized as follows: (i) The motivation and the formulation of the energy saving problem and proposing a minimization solution, (ii) The description of the power-aware routing algorithms aiming at maximizing power saving and establishing a trade-off in terms of network performance, computation complexity and reliability. (iii) The implementation and evaluation of the performance of the power-efficient algorithms under various conditions.

4.2 Motivation and problem formulation

4.2.1 Motivation

Studies of traffic generated by data centers during different periods showed that communication patterns vary depending on the time of usage (e.g. mornings, nights, weekends, working days, holidays)[13]. This means that the traffic does not reach the network's peak capacity except in rush times. It is noted also that a typical data center operates only at 5% to 25% of its maximum capacity depending on the periods of operation[13] [15]. If the load is low, many servers switch to idle state. However, idle state causes a great energy waste, since servers still consume up to 70 % of their peak power[16]. In addition, the traditional routing algorithms do not take into consideration the non-proportionality between the traffic load and the energy consumption which worsens the situation. The work in [81] stated that, at 15 % load, in case of energy proportionality, the network can save 975,000 watts and assuming that the average rate of electricity is $0.07 per kwatt-hour, a data center can save $3.8M over 4 years. Based on these observations, we can deduce that when the network is under-utilized, the traffic can be satisfied by only a subset of devices and the idle ones can be powered off to save the energy which can reduce the cost of the network.
In this context, many proposals are trying to make the power consumption proportional to the traffic workload (see section 2.5.4.5 in chapter 2). The idea is to calculate first the best routes in terms of demand satisfaction[13], bandwidth allocation[77] or throughput[60]. Then, the nodes impacted in these routes are kept active and the others are disabled. However, even though these schemes are solving the problem of non-proportionality of the network and conserving a large amount of energy, they still suffer from long computation time and complexity of routes calculation especially when dealing with a large number of servers. The large computation time can affect the traffic delivery delay and consequently the network performance. In addition, good performance cannot be guaranteed when congestion and failures occur. Based on these challenges, the aim of this chapter is to propose a power aware routing algorithms that contribute in saving the energy while offering a short computation time.

4.2.2 Problem formulation

The objective of any power-aware approach is to deliver the highest performance while minimizing the energy consumed by the network devices. The power consumption of the network devices depends essentially on the configuration of their hardware and the characteristics of the traffic load passing by them[82]. The hardware configuration

of devices consists of two parts: the type of chassis, linecards and cooling machines which have a predetermined and fixed energy consumption, and the second part is the number of enabled ports and the capacity of a port. We suppose, in our work, that all ports have the same capacity. Also, the workload can affect the energy depending on its rate and the size of packets. So, let Ω be the power consumed by cooling machines, fans, chassis and linear-cards in a network device and $n_{(Sw+S)}$ be the number of switches and servers implemented in the data center. Let p be the maximum energy consumed by the ports of a network device when it is fully utilized, P denotes the total power consumed by the data center, Sw and S denote the set of switches and the set of servers, respectively. Table 4.1 presents the notations used in this section.

TABLE 4.1: Summary of notations used in section 4.2.2.

Notation	Description
P	Total power consumed by the data center
$n_{(Sw+S)}$	Number of switches and servers
p	Maximum energy consumption of device ports
Sw	Set of switches
S	Set of servers
Ω	Power consumed by cooling machines, fans, chasis and linecards
ρ	Fraction of power wasted by idle nodes
u	CPU usage
$L(t)$	Set of sender and receiver nodes
R	Routing at a given time t
$(TH(R), AD(R), APL(R), RL(R))$	Resultant average throughput, latency, Apl and reliability level after the routing R
(Th, Ad, Apl, RL)	Thresholds

The power consumed by a data center network is the sum of the power consumed by the ports in addition to the fixed power consumed by the hardware.

$$\begin{aligned} Min \quad P &= Min \quad (n_{(Sw+S)}\Omega + \sum_{i \in Sw} p_i + \sum_{j \in S} p_j) \\ &= n_{(Sw+S)}\Omega + Min \quad (\sum_{i \in Sw} p_i + \sum_{j \in S} p_j) \end{aligned} \tag{4.1}$$

The maximum power consumed by a network device port can be calculated as follows, since it depends on the load fluctuation:

$$p(u(t)) = \rho p_{max} + (1 - \rho) p_{max} u(t) \tag{4.2}$$

Where ρ denotes the fraction of the power wasted by a node when it is idle. u is the CPU usage that changes over the time because of the fluctuation of the workload.
Minimizing the overall power consumption is done by minimizing the power consumed by switches and servers. The power consumed by cooling machines, chasis and linecards will not be considered as it is a fixed part in the network. Since we assumed that all ports have the same capacity, their maximum power consumption will be the same and equal to p_{max}.

$$\begin{aligned} Min \quad P = Min \quad & [\sum_{i \in Sw} (\rho p_{max} + (1-\rho) p_{max} u_i(t)) \\ & + \sum_{j \in S} (\rho p_{max} + (1-\rho) p_{max} u_j(t))] \end{aligned} \tag{4.3}$$

We can remark that even if the network devices are idle ($u(t) = 0$), they still waste a quantity of energy equal to ρp_{max}. Since our aim is to make the energy proportional to the workload, ρp_{max} must be proportional to $u(t)$. It means when $u(t) = 0$, ρp_{max} must be equal to 0. Therefore, we define a variable s as:

$$\begin{cases} s = 0 \quad if \quad u(t) = 0 \\ s = 1 \quad if \quad u(t) > 0 \end{cases}$$

$$\begin{aligned} Min \quad P = Min \quad & [\sum_{i \in Sw} (\rho p_{max} s_i + (1-\rho) p_{max} u_i(t)) \\ & + \sum_{j \in S} (\rho p_{max} s_j + (1-\rho) p_{max} u_j(t))] \end{aligned} \tag{4.4}$$

Let $L(t)$ be the set of sender and receiver servers at a time *t* (servers load). These servers can not contribute in the minimization of power because they certainly accept or send data ($u(t) > 0$).

$$\begin{aligned} Min \quad P = & \sum_{k \in L(t)} (\rho p_{max} + (1-\rho) p_{max} u_k(t)) \\ + Min \quad & [\sum_{i \in Sw} (\rho p_{max} s_i + (1-\rho) p_{max} u_i(t)) \\ & + \sum_{j \in S \setminus L(t)} (\rho p_{max} s_j + (1-\rho) p_{max} u_j(t))] \end{aligned} \tag{4.5}$$

Consequently, for a maximum power saving, we have to maximize the number of devices (switches and servers) with $u(t) = 0$ to switch them off while having as constraint the reliability and high performance of the system. The objective of the guaranteed performance power aware algorithm is to deliver all communication flows based on the traffic matrix at a given time t while the total number of enabled servers and switches is as small as possible and the total power saved is as maximum as possible. This target should be achieved while meeting also a predefined performance thresholds. The key performance metrics used in this chapter are throughput, average delay and APL since they are the most important metrics for time-sensitive data centers and data-intensive

computations. R denotes the routing at a given time t, $TH(R)$ denotes the resultant average network throughput from the routing R at a given time t, $AD(R)$ denotes the average latency of the network, $APL(R)$ denotes the average path length and $RL(R)$ denotes the minimum number of available paths for each flow in the network. We assume that we define Th as a throughput threshold, Ad as delay threshold, Apl as average path length threshold and RL the reliability requirement parameter. The minimisation of the power consumption should be achieved under the constraints of the equations 4.6, 4.7, 4.8 and 4.9.

$$TH(R) \geq Th \tag{4.6}$$

$$AD(R) \leq Ad \tag{4.7}$$

$$APL(R) \leq Apl \tag{4.8}$$

$$RL(R) \geq RL \tag{4.9}$$

4.3 Power aware design

A good routing algorithm should be able to satisfy the requirements of a DCN by delivering the communication with a small latency, using the shortest paths while being traffic aware (dynamic to change routes in case of failures) and with a low computation complexity. In order to design a power-aware routing algorithm, these requirements should always be respected and the performance of the network should be fully offered. The basic idea of our power aware designs is as follows: First, according to the traffic matrix, we define the nodes that should remain active. Second, we compute the throughput, latency and APL using the basic routing of the DCN and deduce the performance thresholds tolerated by the administrator of the network. Third, we remove the unneeded devices in the communication until the performance is adjusted to the threshold. Finally, the communication is delivered using only the subset of the network derived from the previous steps.

4.4 Power saving strategy

As we have noted previously, when there is no traffic passing by an idle network device, there is still a significant amount of consumed energy compared to a fully utilized device. Thus, to save energy, the network should be reduced to only the active components. Generally, there is three ways to reduce the energy consumed by idle nodes. Most of the proposals such as Elastic tree [13] suggest to power off the entire device

including the fixed power consuming part (linecard, chasis, fans, etc). However, when only one port is active in a device, the whole device should be maintained active. In addition, the passage from a totally powered off device to active status takes time and affects the overall performance of the network. The second option is to power off the unused ports in a device. If the device is a switch and all ports are deactivated, it can be totally powered off. The third strategy is to downgrade the ports to a lower rate. It is used when studying the queue in each port and fixing the rate according to the state of the buffer. Even if it showed interesting results, this method is far from being the optimal one because powering off the ports ensures more energy saving. In our work, we will adopt the second option because it offers the largest amount of power conservation and it is more adequate to our design.

4.5 Power aware routing algorithm

The power-aware routing algorithms presented in this chapter are composed of three modules: Choosing the nodes to disable, fixing thresholds and disabling network devices and performance adjustment. The inputs of these algorithms are T_0 which denotes the data center topology, TM that represents the traffic matrix and indicates all nodes sending or receiving a communication at a given time t, TP which is the threshold percentage or the performance decrease tolerated by the network administrator after disabling some ports of servers and switches and RL is the reliability level. The thresholds are chosen by the administrator of the network. In fact, before implementing the algorithm in a real data center, the administrator needs to gather information about the network topology, the traffic and the applications installed in the data center. For example, since some applications are not sensitive to latency such as MapReduce, openssl, flight-search[83], the applications owners can sacrifice a little bit in terms of latency in order to reduce the cost. In this case, they can indicate how sensitive to time their applications are. In addition, the administrator should study the behaviour of the network for a period of time, which can be done using some existing network technologies and protocols such as OpenFlow[84] and NetFlow[85]. In this way, the administrator can decide the reliability restriction by studying the congestion periods (where the network needs more backup routes to ensure the load balancing), the low load periods time (one route is enough to maintain the required performance) and the percentage of failures of the infrastructure (since failures in data centers are rare events). The output of the power-aware algorithm is the performance of the final topology ($Th1$: the final throughput, $Ad1$: the final latency, $Apl1$: the final average path length), PD is the set of deactivated nodes, EN is the set of active nodes. T_1 is the new topology after disabling nodes. (Th; Ad; Apl) denote the performance parameter thresholds and AS denotes the

number of servers to re-activate or deactivate. The Table 4.2 represents the summary of all notations used in the routing algorithms.

TABLE 4.2: Summary of notations used in the routing algorithms.

Notation	Description
T_0	Data center topology
TM	Traffic matrix
TP	Threshold percentage
IS	Nodes of the traffic matrix
PD	Nodes to disable
EN	Active nodes
Th, Th_0, Th_1	Throughput threshold, initial throughput, final throughput
Ad, Ad_0, Ad_1	Latency threshold, initial latency, final latency
Apl, Apl_0, Apl_1	APL threshold, initial APL, final APL
T_1	New topology
AS 1,2,3	Number of servers to re-activate or deactivate.
RL	Reliability level

The Flowchart in Figure 4.1 describes the modules of the power aware routing algorithms.

4.6 Power saving evaluation

The amount of saved energy depends on several conditions: The most important one is the reliability factor. As multi-paths between nodes should be available, more unneeded devices should be maintained active. The traffic pattern, the system load, the size of the network and the tolerance of the system to the performance loss are other factors that affect the energy conservation. Moreover, reducing the energy consumption depends also on reducing the energy consumed by the fans, linecards, chasis etc. This part of power conservation will not be treated in our calculation as indicated in a previous section. The percentage of energy saved ($\%ES$) will be calculated as follow:

$$\%ES = \frac{\sum P_{disabled} * 100}{n_{sw} \sum P_{sw} + n_s \sum P_s}, \tag{4.10}$$

Where, n_{sw} is the total number of switches and n_s is the total number of servers in the network. $\sum P_{disabled}$, $\sum P_{sw}$, $\sum P_s$ denote respectively the total number of disabled ports, the number of ports per switch and the number of ports per server.

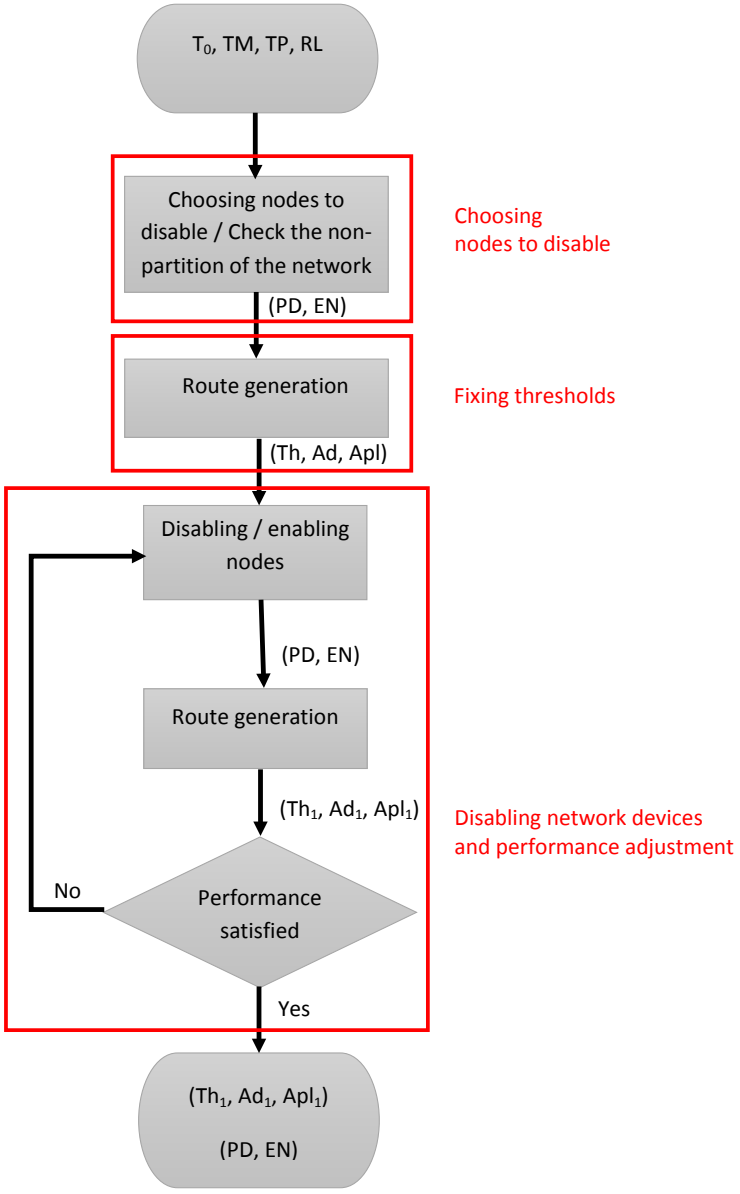


FIGURE 4.1: Modules of the power-aware routing Algorithm.

4.7 Power aware routing algorithm for PTNet data center

4.7.1 PTNet power aware routing algorithm

The first power aware routing algorithm is designed to make PTNet a power efficient topology[86]. As we discussed in section 4.2.1, the proposed power aware algorithms suffer from calculation complexity because of routes searching. Therefore, our solution will be to define only the sender and receiver nodes without wasting time to search for the intermediary nodes (which contribute to the routing of packets). A minimum set of nodes is kept active (the senders, receivers and some calculated nodes to maintain the

network fully connected and the number of disjointed paths equal to 2). Then, ports of other devices are switched off. Afterwards, we compare the performance of the original network, in terms of throughput, latency and average path length to the power aware network. If the new performance is below a fixed threshold, some server ports will be activated randomly. The power aware routing algorithm is described in Algorithm 6.

Algorithm 6 Power-aware routing algorithm

Input (T_0, TM, TP, RL**)**
T_0: PTNet initial topology
TM: The traffic matrix
TP: Threshold percentage
RL: Reliability level
Output ($Th_1, Ad_1, Apl_1, PD, T_1$**)**
Th_1, Ad_1, Apl_1 (Final throughput, Final latency, Final APL),
PD (Vector of deactivated nodes),
T_1 (New topology after disabling nodes)
Choosing nodes to disable
IS =GetNotReceivingSendingServers (T_0, TM)
PD =GetPossibleDisabledNodes (T_0, IS)
EN =GetActiveNodes (T_0, PD)
Fixing thresholds
(Th_0, Ad_0, Apl_0) =ShortestPathBasedRoutingAlgorithm (T_0, TM)
Th = C (Th_0, TP)
Ad = C (Ad_0, TP)
Apl = C (Apl_0, TP)
Disabling network devices and performance adjustment
T_1 =DisableNodes(PD)
attempt = 0
Do
if attempt > 0 **then**
  AS1 = $(Th - Th_1)/((Th_0 - Th_1)/PD)$
  AS2 = $(Ad - Ad_1)/((Ad_0 - Ad_1)/PD)$
  AS3 = $(Apl - Apl_1)/((Apl_0 - Apl_1)/PD)$
  AS = max(AS1,AS2,AS3)
  for i=1..AS **do**
    (T_1, PD) = EnableNodes (PD(i))
  end for
end if
(Th_1, Ad_1, Apl_1) =ShortestPathBasedRoutingAlgorithm (T_1, TM)
attempt = attempt+1
While ($Th_1 < Th$) and ($Ad_1 < Ad$) and ($Apl_1 < Apl$)
return ($Th_1, Ad_1, Apl_1, PD, EN$)

4.7.1.1 Choosing nodes to disable

A "Redundancy and no-partition guaranteed heuristic" is designed to define the nodes to maintain active. This heuristic builds a "minimal network" from the critical nodes and switches. Critical nodes represent elements of the architecture that can partition a topology if they are not active. The heuristic will determine the nodes that can be deactivated (a deactivated node is a node that has all its ports switched off.).
Figure 4.2 shows that a server can be reachable from 2 disjointed paths: either passing by one of the servers connected to the same switch (including the master server) or passing by the master server in the homologue switch. The master server is a critical node that plays the role of a relay transmitting packets to or from the two sides of communication. Thus, a master server that does not receive neither send data will not be deactivated if one of the servers connected to the same switch is active. In addition, if one of the servers from an homologue switch and connected to the master is active, the master must stay active. In this way, a node can keep its two disjointed paths available ($RL = 2$). Thus, a non-active master server (i,j,i) is not added to the vector PD unless all servers connected to it or to the switch (i,j) are not active. Active masters in the same cell will stay connected to guarantee a maximum performance.

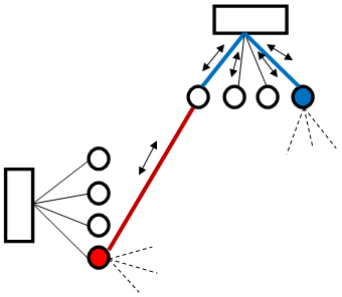


FIGURE 4.2: The importance of the master server in PTNet.

$$TM = \begin{bmatrix} 2 & 5 \\ 6 & 12 \\ 2 & 15 \\ 16 & 5 \end{bmatrix}$$

For the sake of simplicity, we assume that TM is the traffic matrix of a 16-servers PTNet where the first column represents the senders and the second represents the receivers. Figure 4.3(a) shows the nodes that must be active according to TM (only senders, receivers and masters). In this example, the network is partitioned and nodes are not fully connected. A way to ensure that the network will not be partitioned is to connect all masters (i,j,i) to the server (i-1,j,i) using Algorithm 7. By applying this heuristic,

the network depicted in Figure 4.3(a) is not anymore partitioned and the solution is presented in Figure 4.3(b). Figure 4.4(a) shows the minimum number of servers that must stay active in a 16-servers PTNet when all masters must be active. Figure 4.4(b) shows the minimum number of active servers when one master can be deactivated.
In this phase, we assume that the performance is not below the tolerated threshold and the system can find the “best" routing paths. The goal here is to compute only the maximum number of servers that can be deactivated without partitioning the network, and therefore a minimum of computation time can be achieved since there is no need to search for routing paths between senders and receivers.
To summarize, in this phase, the vector EN will contain only the senders, the receivers and the nodes calculated by applying the "Redundancy and no-partition heuristic". It means routes will not be calculated and included.

Algorithm 7 No-partition guaranteed heuristic

Input: n (Number of ports per switch), s (Number of switches per cell), PD (Vector of possible deactivated nodes), node (List of nodes)
Output: node (Updated list of nodes), PD (Updated vector of possible deactivated nodes)

```
for i = 1..n do
  for j = 1..s do
    if node (i, j, i) ∉ PD then
      k = i
      Do
      k=k-1
      if k ≤ 0 then
        k=n
      end if
      While node (k, j, k) ∈ PD
      if node (i, j, i) and node (k, j, i) not linked then
        Activate link (node (i, j, i), node (k, j, i))
        Remove node (k, j, i) from PD
      end if
    end if
  end for
end for
```

4.7.1.2 Fixing thresholds

In this step, all devices are still active. The basic routing algorithm of PTNet (see chapter 3, section 3.3.2.3) is run and the initial performance of the network is estimated. Then, based on TP, the performance threshold is calculated. The APL, the average delay and the average throughput are the metrics considered in our system to evaluate the

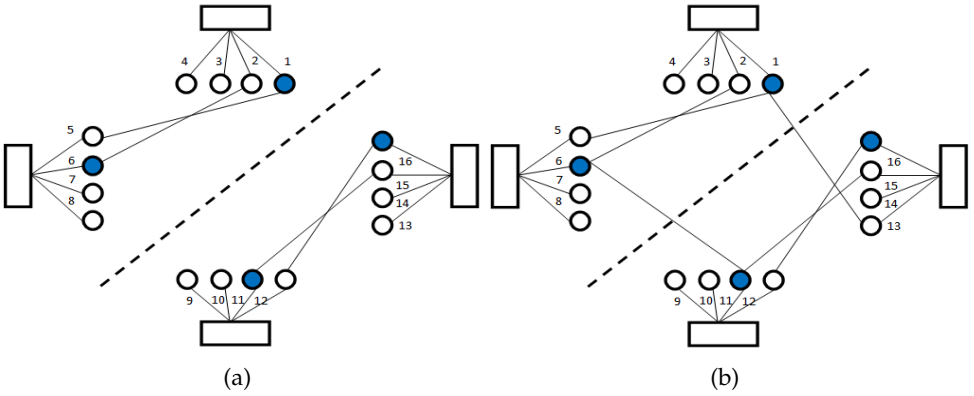


FIGURE 4.3: Guaranteeing a non-partitioned network.

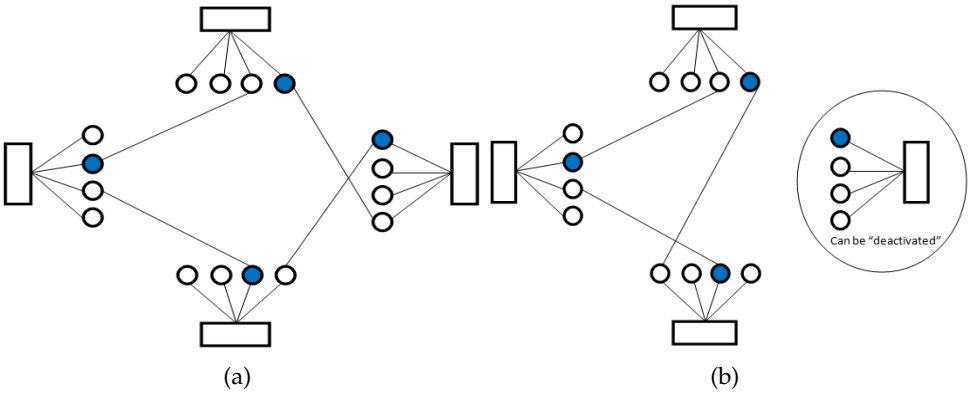


FIGURE 4.4: Minimum number of active nodes.

performance. When disabling the nodes, the performance should not be under the threshold.

4.7.1.3 Disabling network devices and performance adjustment

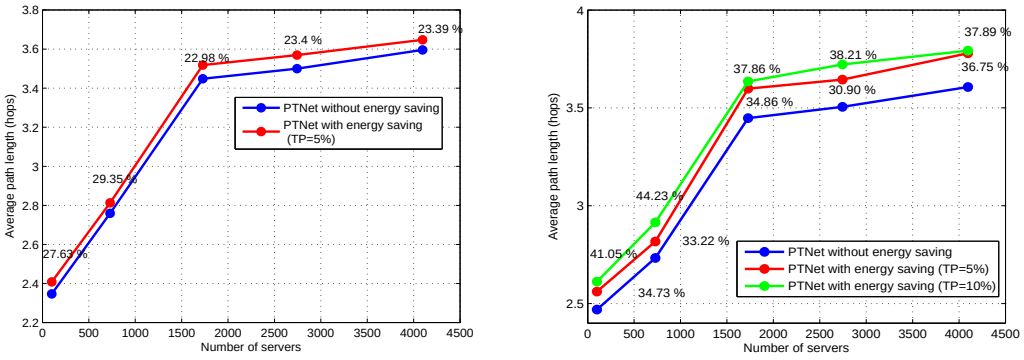
After applying the shortest path algorithm, performance parameters of the original network as well as the performance threshold tolerated by the network administrator are computed. Then, all ports of nodes in the vector PD and their correspondent ports of switches are disabled. A switch will not be totally shut down unless all its ports are deactivated. The next step is to re-apply the shortest path routing algorithm: if the active nodes do not manage to achieve the required performance threshold, then some other nodes will be activated randomly. This step is to be repeated until regaining the performance threshold and the algorithm will return the vector of deactivated devices PD.

4.7.2 System evaluation

Several experiments were carried out to evaluate the first power aware routing algorithm such as the evaluation of the APL, the diameter, the throughput and the latency. PTNet's power aware routing is compared to the original routing of PTNet and other data center networks (BCube, DCell and Flatnet) operating with their non-power-aware routing algorithms. The configuration of the experiments is the same as the configuration summarized in Table 3.9 in chapter 3.

4.7.2.1 Average path length

The APL is computed for a some-to-some pattern to evaluate the network in a realistic environment. A traffic distribution is randomly created after fixing a percentage of nodes sending or receiving transmissions. Figure 4.5(a) shows the difference between



(a) (b)

FIGURE 4.5: Average path length

the APL of a PTNet without deactivating any device and the APL of the power-aware PTNet when 40% of nodes are idle (not sending or receiving packets) and the threshold percentage TP is equal to 5%. While preserving the performance of the system, the green PTNet saves up to 29% of the total energy consumed. In the experiments shown in Figure 4.5(b), we used a traffic distribution where 60% of the servers are idle, and we have compared the results given by PTNet without energy saving to the one applying the power aware routing algorithm. The threshold percentages are fixed to 5% and 10%. We can notice that it is up to the network administrator to set up a trade-off between performance and power saving. When he/she increases the tolerated threshold, the performance decreases however more energy is saved. If performance cannot be sacrificed, then obviously the energy consumption will increase. The percentage of energy saved is calculated as indicated in equation (4.10) in section 4.6.

4.7.2.2 Network average latency

After proposing a power-aware routing algorithm and deactivating nodes, we evaluated the network average latency. A threshold TP of 5% then 10% is fixed to maintain better performance than the other data center networks which is depicted in Figure 4.6(a). A bigger threshold contributes to save more energy, however, the performance can be worsened and underperform the other topologies. With this configuration, the energy is saved up to 44% of the total consumed energy if initially there is 60% of servers not sending nor receiving energy. Figures 4.6(b) and 4.6(c) show respectively the energy saved (percentages included in the graph) when 40% and 60% of servers are not sending nor receiving data vs the performance.

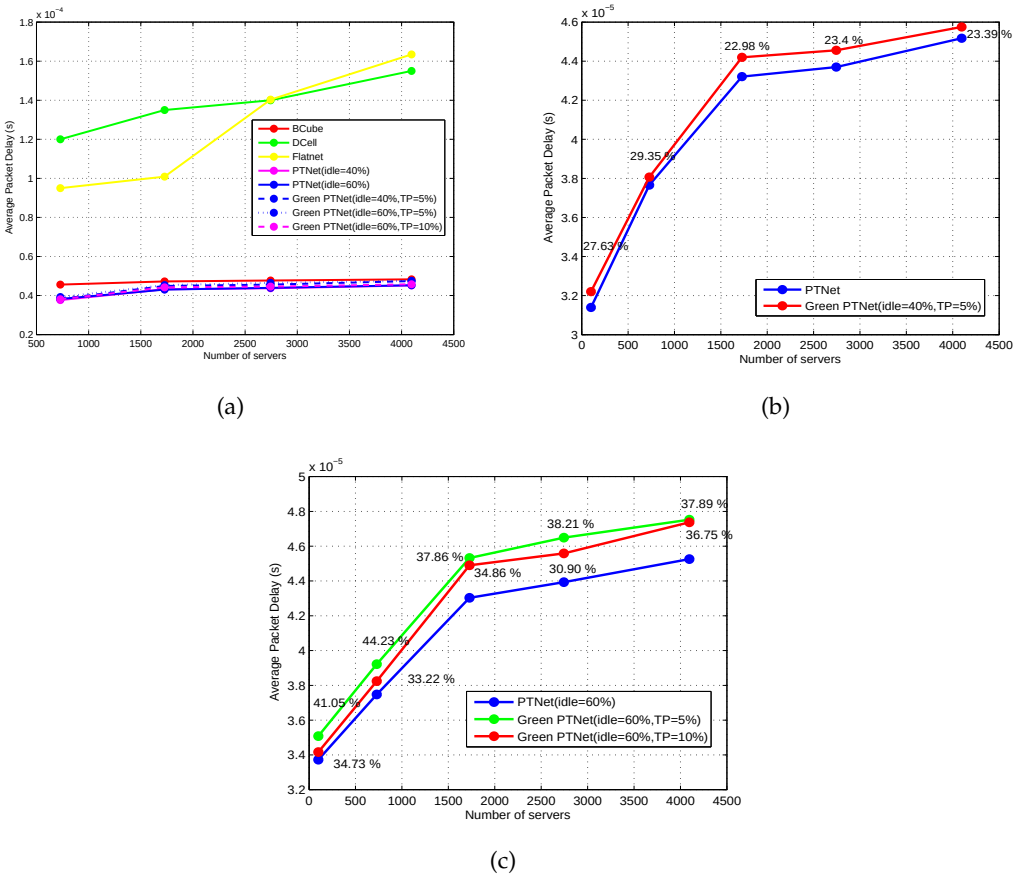
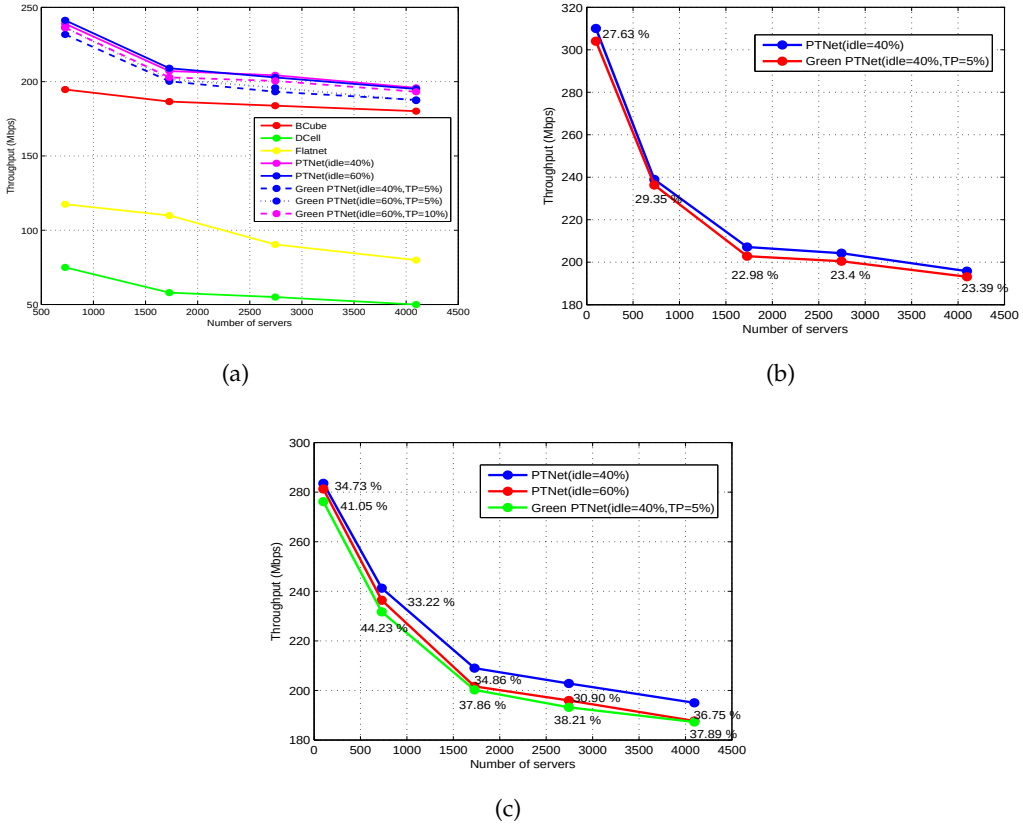


FIGURE 4.6: Average packet delay.

4.7.2.3 Throughput

Figures 4.7(b) and 4.7(c) show that the use of the proposed power-aware routing algorithm helps to save energy with a negligible performance degradation, especially when a small tolerated threshold is used. Obviously performance degradation leads to more

energy saving. In addition, Figure 4.7(a) proves that, with a TP equal to 5% and 10 %, our system still outperforms the non-power-aware systems.



(a) (b)

(c)

FIGURE 4.7: Average throughput.

4.7.2.4 Performance under faulty conditions

For the case of 40% of nodes not sending or receiving data, we chose randomly a set of faulty servers and we evaluated the impact on performance. If the failed nodes are from the deactivated ones (idle nodes), the performance is not affected. In the experiments shown in Figure 4.8, we have failed servers chosen from the active nodes and the tolerated performance decrease TP must not exceed 5%. The results show that energy saved after 10% of failed servers is still good despite re-activating some nodes to maintain an appropriate performance level. This is explained by the fact that the nodes managed to find other routes among the active nodes which does not oblige the system to re-activate many additional nodes. However, the performance decreased since the new routes are not the best ones.

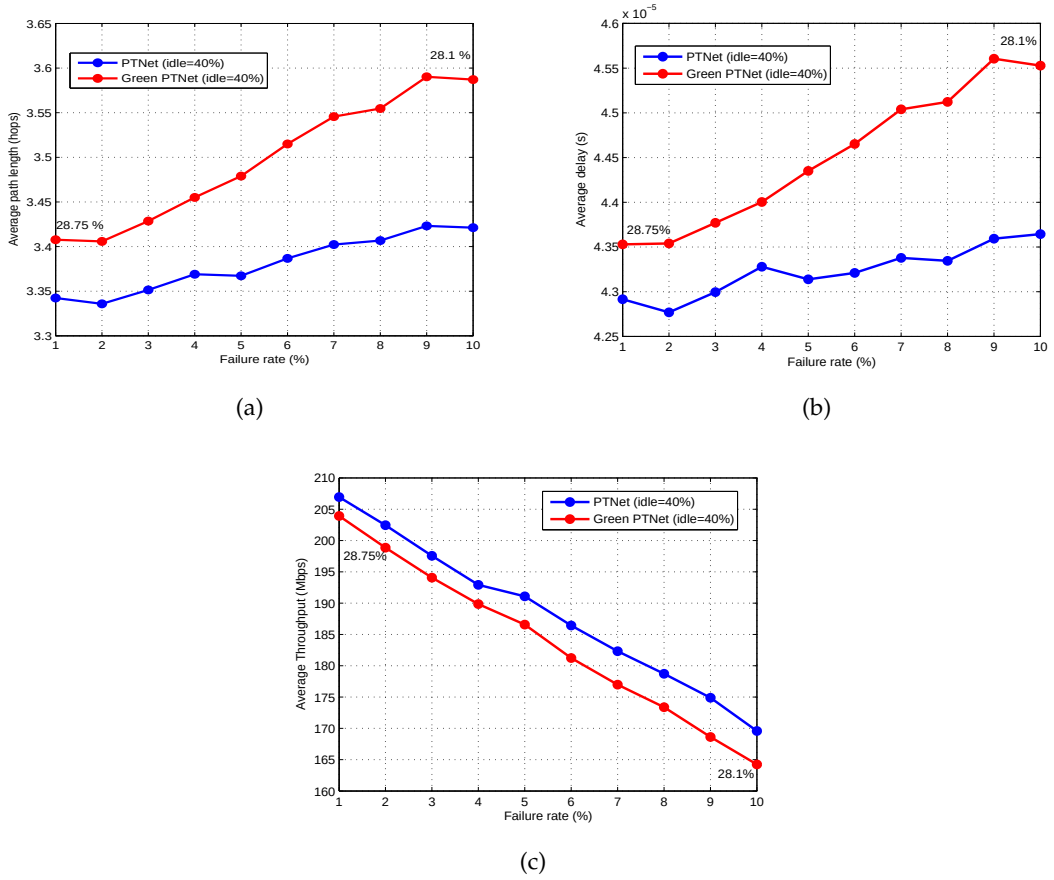


FIGURE 4.8: The performance of a 1200-servers of the PTNet vs Green PTNet under various faulty conditions (Some-to-some communication pattern).

4.7.2.5 Power consumption

The proposed power-aware routing algorithm achieves a trade-off between saved energy, good performance and computation time. Figure 4.9 shows the energy saved of 1200-servers PTNet ($s = 3$ and $n = 20$) for different loads and a threshold percentage of 5%. Results reveal that achieving 10-40% is possible, and by augmenting the threshold percentage, power saving can increase. Moreover, the percentage of servers not receiving nor sending data has substantial impact on power conservation. The system saves more energy when the network load is low (i.e. high percentage of idle nodes). We can also notice that when the load is very low (i.e. starting from 70 % of idle servers), the power consumption slightly drops and becomes constant. This can be explained by the fact that, at a low load, most of servers will be deactivated in the first step. Consequently, the performance will drop. Since we have fixed 5% of tolerance, many servers will be randomly re-activated until regaining good performance. A random choice of nodes causes the re-activation of a big number of servers. Hence, the energy saved is not important.

Even though the energy saving is not maximum in a low load, 35% of gain is still a good result. Compared to existing approaches, one of the biggest advantages of the proposed algorithm is the computation efficiency: existing approaches have exponential time complexity due the large searching space of all nodes contributing in the communication especially in huge data centers. However, the proposed algorithm achieves the traffic communication by just the sending and receiving servers extracted from the traffic matrix and some extra nodes computed in a negligible time with a little performance degradation.

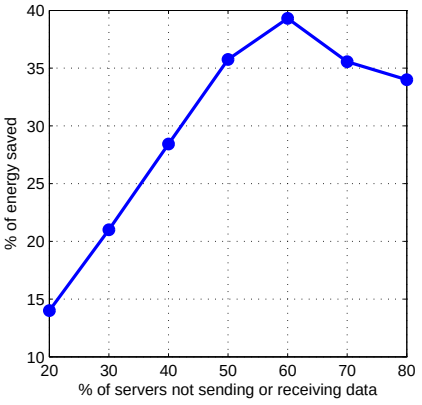


FIGURE 4.9: Power saved by the power aware routing algorithm of PTNet.

In this first power aware routing algorithm, we are re-activating the nodes randomly to adjust the performance of the network. Even though this approach helps to gain a large amount of energy, it is not an optimal solution. The nodes to deactivate or re-activate must be studied in advance. To handle this problem, a power aware routing algorithm based on vital nodes is proposed.

4.8 Power aware routing algorithm based on vital nodes

Searching all the routes in real time as done in many proposed approaches can be expensive in terms of computational complexity and time. Also, de-activating nodes without studying those contributing in the communication routes as described in the previous section can save energy, however, it is not an optimal approach. Hence, instead of calculating the nodes required to transport data flows in real time, we propose to calculate the vital nodes contributing in the transportation of the traffic between different clusters in the network and classify them by importance[87][88]. This task is accomplished during the network initialization phase. In this way, we will not waste time in searching the routes in every traffic pattern. Indeed, the pre-calculated most

important nodes will be just selected and kept active in addition to the sender and destination nodes.
The proposed routing algorithm is described as follows: First, according to the traffic matrix, we define the vital nodes (described in section 4.8.1). Second, we compute the throughput, latency and APL using the basic routing of the DCN and deduce the performance threshold tolerated by the administrator of the network. Third, we remove the non vital devices until the performance is adjusted to the threshold. Finally, the communication is delivered using only the subset of the network derived from the previous steps.

4.8.1 Calculate vital nodes

Nowadays, data centers are becoming larger, more complex and more vulnerable to attacks and failures and it is clearly recognized that the survivability of the network is a critical issue. To understand what are the vital nodes in a data center, we will assume that every packet in the network is routed through the shortest path in the communication graph. When a node in this route fails (broken down or deactivated), we have to replace the old path by a new one, preferably by the shortest path and necessarily it does not contain the disabled node. The new route is longer than the previous one and it will not give the optimal performance of the network. However, in some cases, the failure of a node will cause the partition of the network and the communication will be lost. Thus, it is important to know which disabled nodes causing the degradation of the performance or the cut of the communication and which nodes are most engaged in the traffic load. Such nodes are called *vital nodes*.

4.8.1.1 Rules to select vital nodes

Intuitively, we will define five rules that describe the vital nodes in a data center. These rules will not cover completely the concept of vital nodes because many factors are contributing to classify nodes by importance such as betweenness, closeness, degree, eigenvector centrality, mutual-information, local clustering coefficient, etc. However, we will choose the ones that seem to be essential.
Rule 1: The node that is connected to a larger number of neighbor nodes would have a larger influence in the network. That is, if a node a in a network has more connected neighbors than a node b, then a is more important than b.
Rule 2: If a node contributes once or more in the traffic communication, it is considered as a key node (we assume that communications in a data center are delivered through shortest paths). Thus, if a node a is impacted in more communication flows than a node

b, *a* is more important than *b*.
Rule 3: A node which is close to a maximum number of destinations is considered as an important node. A node *a* that has a smaller number of hops to reach other nodes is more important than a node *b* with a larger number of hops.
Rule 4: If disabling a node disconnects the network and split it into two or more disconnected components, this node is the most important node in the network and it is considered as a vulnerable point and an indispensable node to design a reliable network.
Rule 5: if a node *a* and a node *b* have similar positions and the similar impact in the network according to the previous rules, *a* and *b* have an equivalent importance.

4.8.1.2 Formalizing the rules

To rank nodes by importance, three factors are taken into consideration which are the degree, the closeness and the betweenness. Vital nodes are calculated in an all-to-all communication between different clusters in the network. Thus, we divide the network into c clusters, each cluster contains n nodes. The array of vital nodes will be computed between source clusters C_s and destination clusters C_d. We abstract the network using the graph theory. The interconnection of nodes in the network is represented by an undirected graph $G(V, E)$, where V is the set of nodes, m is the size of V and E is the set of edges. The factors that determine the importance of the nodes are explained as follows:
-Degree of the nodes: Rule 1 involves the degree as the measure index. The degree is defined as the number of edges incident upon a node v_i. It is considered as the base of the node importance since it reflects the ability of the node to directly obtain network flow content.

$$d_{v_i} = \sum_{j=1}^{m} \delta(v_j, v_i), \tag{4.11}$$

where v_i and $v_j \in V$. $\delta(v_j, v_i)$ is equal to 1, if v_i is connected to v_j and it is equal to 0 otherwise.
-Betweenness: Rule 2 can be measured by the betweenness which is described by the following expression:

$$b_{vi} = \sum_{j}^{n} \sum_{k}^{n} \frac{\delta_{jk}(v_i)}{\delta_{jk}}, \tag{4.12}$$

where, j is a node from the cluster source C_s, k is a node from the cluster destination C_d, δ_{jk} is the total number of shortest paths between a node j from C_s and nodes k from C_d and $\delta_{jk}(v_i)$ is the number of these paths that go through v_i.

The betweenness reflects the capacity of a node to provide the maximum shortest paths in the communication tasks. It is also used to determine the location of the heavy workload in the network.

-Closeness: Rule 3 is deployed by the closeness. The closeness of a node v_i is defined as the mean shortest path between v_i and all other nodes v_k of C_d reachable from it.

$$c_{vi} = \frac{n^2}{\sum_{k=1}^{n} d(v_i, v_k)}, \tag{4.13}$$

where $d(v_i, k)$ is the shortest distance between v_i and v_k.

Closeness can be regarded as a measure of how far the information can spread from a given node to other reachable nodes in the network.

The algorithm 8 shows how to compute the set of importance indexes when an all-to-all communication is established between two clusters C_s and C_d. S_{cd} is the set of nodes participating in this communication.

The resultant matrix of vital nodes and correspondent indexes is given by:

Algorithm 8 Calculation of importance indexes

Input: C_s (Cluster source), C_d (Cluster destination),
S_{cd} (Set of nodes included in the communication), d (Degree),
b (Betweenness), c (Closeness), v (Node $\in V$)
Output: VN (Matrix of vital nodes)
VN=[ ]
for $C_s = 1..c$ **do**
  for $C_d = 1..c$ **do**
    $VN(C_s, C_d)$=[ ]
    for $v_i \in S_{sd}$ **do**
      calculate d_{vi}
      calculate b_{vi}
      calculate c_{vi}
      add $[vi, d_{vi}, b_{vi}, c_{vi}]$ to $VN(C_s, C_d)$
    end for
  end for
end for

$$VN(C_s, C_d) = \begin{bmatrix} vi & d_{vi} & b_{vi} & c_{vi} \\ \vdots & \vdots & \vdots & \vdots \\ vj & d_{vj} & b_{vj} & c_{vj} \end{bmatrix}$$

Three parameters are used to determine the vital nodes. This may be conflicting because the evaluation of nodes importance in a complex network such as data center should be based on a single index. Research on multiple objective optimisations brought to light the Principal Component Analysis (PCA)[89] which is used to reduce

the dimension of the data and restrict the importance of the nodes into one index. After applying the PCA, nodes are ranked by respecting the rules 1,2,3 and 5 to conclude finally the node importance sequence of the entire network. The matrix of vital nodes importance VNI is as follows:

$$VNI(C_s, C_d) = \begin{bmatrix} vi & I_{vi} \\ \vdots & \vdots \\ vj & I_{vj} \end{bmatrix}$$

To obtain VNI, we considered that all rules have the same importance. In fact, their order depends on the applications installed on the data center and the data center requirements. For example, if the requirement of the application is to meet time constraints and reduce the latency, closeness should be the most important rule. Therefore, rules should be ordered according to the application needs. However, in this work, we wanted to be general and do not target any specific application. In addition, we chose the five rules that are indispensable for most of the data centers.
Vital nodes are calculated between each two clusters and not between any two nodes in order to reduce the size of VNI and store the minimum number of nodes which accelerates the search of nodes to keep active in a traffic load (section 4.8.2.1).
-Articulation points: A node v_i in a graph G is an articulation point (known also as a cut node), if removing it (cut all its edges) splits the graph into one or more components. Any network with an articulation point is considered a fragile network because disabling this single node causes the loss of the connectivity between nodes. A graph is called biconnected, if it does not contain any articulation point which is the best case. The simplest approach to define the articulation points in a graph is to remove the nodes one by one and see if the removal of one of them causes the partition of the graph. The following steps determine if a node $v_i \in V$ is an articulation point:

- Remove the node v_i.
- Check if the graph G is partitioned using the DFS (depth-first search) algorithm[90].
- Reconnect v_i to the graph.

The DFS is an algorithm to search into a graph data structure. It consists of choosing arbitrary a root node from the graph, exploring as far as possible into all branches and then checking if all nodes are visited. The algorithm 9 shows how to use DFS scheme to check the status of the network (partitioned or not). The search of articulation points is covered by the rule 4. These nodes are the most critical points and their failure means the cut of the network into two or more parts that can not communicate. The time

Algorithm 9 DFS: Depth-First search algorithm

Input: V (Set of nodes), v (Node $\in V$), G (Undirected graph)
choose v_j randomly from V
visited =[ ]
Function DFS (V, v_j)
if $v_j \notin$ visited **then**
  add v_j to Visited
end if
for v_k connected to v_j **do**
  if v_k is parent **then**
    DFS (V, v_k)
  else
    add v_k to Visited
  end if
end for
End Function
if all nodes $\in V$ are visited **then**
  G is not partitioned
else
  G is partitioned
end if

complexity of the above method is O(V+E) which is acceptable.

As an example, we applied the rules for a FiConn[32] topology described in chapter 2.

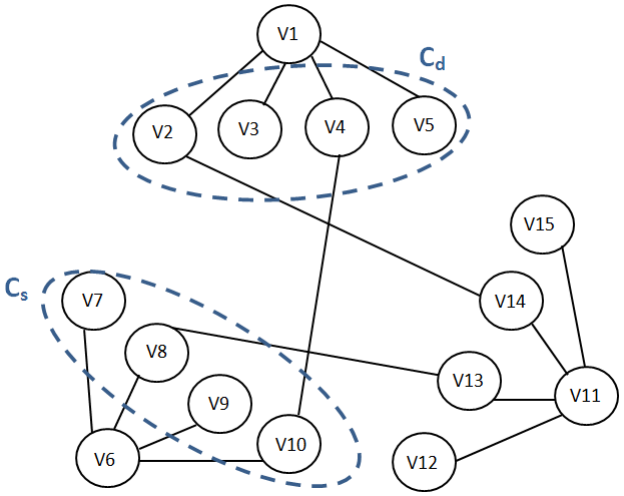


FIGURE 4.10: Clusters of FiConn data center topology (n=4).

A 2-layer FiConn is characterized by its simple connection pattern. Thus, for simplicity reasons, we chose to use it for demonstration as shown in Figure 4.10. FiConn will be divided into clusters. Each cluster contains 4 servers. let C_s be the cluster source and C_d be the cluster destination. For an all-to-all communication between the two clusters, we evaluated the importance of the nodes contributing in the communication. The results in table 4.3 shows the rank of each node in the traffic load between C_s and C_d. we can

TABLE 4.3: Importance evaluation.

Nodes	Betweenness	Closeness	Degree	PCA
v2	-	-	-	-
v3	-	-	-	-
v5	-	-	-	-
v7	-	-	-	-
v8	-	-	-	-
v9	-	-	-	-
v1	0.6875	4	4	5.6708
v6	0.6875	1.1429	4	3.4887
v11	0.0625	1.1429	4	3.3982
v4	0.6875	2.6667	2	3.3944
v14	0.0625	1.6	2	2.4892
v10	0.6875	1.6	2	2.5797
v13	0.0625	0.9412	2	1.9860
v12	-	-	-	0
v15	-	-	-	0

see that the two nodes v_{12} and v_{15} do not participate in the communication and their importance is counted as 0. Also, the importance of a node sending or receiving the traffic (nodes of C_s and C_d) is not calculated only if it is an intermediate node. v_1 is the most important node because almost all communications pass through it and it is the closest to the destinations. The matrix $VNI(C_s, C_d)$ is equal to:

$$VNI(C_s, C_d) = \begin{bmatrix} v_1 & 5.6708 \\ v_6 & 3.4887 \\ v_{11} & 3.3982 \\ v_4 & 3.3944 \\ v_{14} & 2.4892 \\ v_{10} & 2.5797 \\ v_{13} & 1.986 \end{bmatrix}$$

For example, assume a malfunction in a node v_{10}, we use the DFS algorithm to see the articulation points in the new graph (the node v_{10} is removed from the graph). As shown in Figure 4.11, v_2, v_8, v_{13} and v_{14} are critical points and without them the network is partitioned. So, in case of power conservation by powering off the unneeded nodes, these nodes should not be deactivated and should be ranked the firsts in the matrix of important indexes.

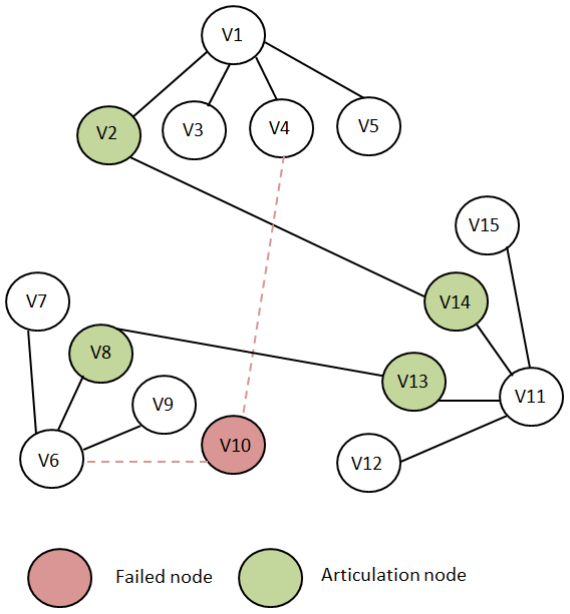


FIGURE 4.11: Articulation nodes in FiConn in case of v_{10} failure.

4.8.2 Vital nodes power aware routing algorithm

4.8.2.1 Choosing nodes to disable

The role of this module is to restrict the network to the minimum number of active devices which includes only the sending and receiving nodes (nodes of the traffic matrix) in addition to the vital nodes participating in the communication and the articulation nodes that without them the network becomes partitioned and loses its reliability. In fact, at a given time t, when having the traffic matrix, we can determine the nodes source, destination and their correspondent clusters. Consequently, we can obtain the matrix VNI of vital nodes for each communicating clusters. Then, these nodes are regrouped and ranked in a matrix RN.

$$RN = \begin{bmatrix} vi & O_i & I_i \\ \vdots & \vdots & \vdots \\ vj & O_j & I_j \end{bmatrix}$$

When a node vi occurs multiple times, its maximum importance is denoted as I_i and O_i indicates its number of occurrences. Nodes that do not contribute in all communications will have 0 as occurrence and importance.
To obtain the unique index of importance, we apply again the PCA to reduce the matrix

RN to:

$$RN = \begin{bmatrix} vi & I_i \\ \vdots & \vdots \\ vj & I_j \end{bmatrix}$$

RN is the matrix of relevant nodes of all communications in the traffic matrix ranked by importance. The array PD (in Algorithm 10) contains the nodes with importance equal to zero and the nodes with a negligible importance ($I_i \cong 0$). Before adding the nodes to PD, they must be checked by the algorithm DFS (Algorithm 9). If one of them is a cut node, it should be added to the array EN in the head of the list. In this way, the minimum subset of the network needed to fulfill the communication is constructed.

4.8.2.2 Fixing thresholds

This module is similar to the first power aware routing algorithm. First, all devices are active. The basic routing algorithm of the DCN is run and the initial performance is estimated. Then, the threshold performance is calculated, according to the threshold percentage TP.

4.8.2.3 Disabling network devices and performance adjustment

The first step in this module is to disable the nodes in the array PD calculated in the first module. Then, we recompute the performance of the system and compare it to the threshold fixed by the administrator. If it is not achieved, some disabled nodes should be reactivated. Indeed, nodes with larger importance are the first candidates. In the other hand, if the new performance is very near to the initial performance, the less important nodes are disabled. To ensure the reliability of the system, the articulation nodes should remain always active. This step is fulfilled by applying DFS algorithm. In this way, the performance is adjusted to guarantee a maximum amount of saved energy. Ranking the nodes by importance enables the system to adjust the performance rapidly.

4.8.2.4 Reliability

In order to maximize the energy saved in a data center, the module "Disabling network devices and performance adjustment" is aggressively reducing the redundancy of the rich connection of the DCN. It means the idle nodes in the redundant paths between servers are turned off. However, even though restricting the DCN to the minimum

Algorithm 10 Power-aware routing algorithm

Input (T_0, TM, TP, VNI, RL**)**
T_0: Initial topology
TM: the traffic matrix
TP: threshold percentage
VNI: Matrix of vital nodes importance
RL: Reliability level
Choosing nodes to disable
IS =GetReceivingSendingServers (T_0, TM)
RN =GetRelevantNodes (TM, VNI, RL)
PD =GetPossibleDisabledNodes (T_0, IS, RN)
EN =GetActiveNodes (T_0, PD)
Fixing thresholds
(Th_0, Ad_0, Apl_0)=RoutingAlgorithmGeneration (T_0, TM)
Th = C (Th_0, TP)
Ad = C (Ad_0, TP)
Apl = C (Apl_0, TP)
Disabling network devices and performance adjustment
(T_1, PD) =DFS(PD)
T_1 =DisableNodes(PD)
attempt = 0
Do
if attempt > 0 **then**
  if ($Th_1 < Th$) and ($Ad_1 > Ad$) and ($Apl_1 > Apl$) **then**
    AS1 = $(Th - Th_1)/((Th_0 - Th_1)/PD)$
    AS2 = $(Ad - Ad_1)/((Ad_0 - Ad_1)/PD)$
    AS3 = $(Apl - Apl_1)/((Apl_0 - Apl_1)/PD)$
    AS = max(AS1,AS2,AS3)
    (T_1, EN) = ChooseNodesToEnableByImportance(PD, AS)
    T_1 =EnableNodes(EN)
  end if
  if ($Ad_0 \cong Ad_1$) and ($Apl_0 \cong Apl_1$) and ($RL = 1$) **then**
    AS1 = $(Ad - Ad_1)/(Ad_1/EN)$
    AS2 = $(Apl - Apl_1)/(Apl_1/EN)$
    AS = max(AS1,AS2)
    (T_1, PD)=ChooseNodesToDisableByImportance(ED, AS)
    (T_1, PD) = DFS(PD)
    T_1 = DisableNodes(PD)
  end if
end if
(Th_1, Ad_1, Apl_1) =RoutingAlgorithmGeneration (T_1, TM)
attempt = attempt+1
While[($Th_1 < Th$) and ($Ad_1 > Ad$) and ($Apl_1 > Apl$)] || [($Ad_0 \cong Ad_1$) and ($Apl_0 \cong Apl_1$) and ($RL = 1$)]
return ($Th_1, Ad_1, Apl_1, PD, EN$)

subset of the network and powering off the idle devices seems to be promising, it is hard to maintain the fault tolerance of the system. In fact, the robustness of the network

requires to keep the maximum number of devices active to guarantee more redundant paths. Consequently, there must be a trade-off between energy conservation and fault-tolerance. To address this issue, we should introduce an additional reliability to the power aware system. For this reason, vital nodes should include the important nodes in the best path in addition to the backup disjointed paths.
If the administrator of the system chooses a reliability level superior to 1 ($RL > 1$), the vital nodes calculated for the backup disjointed paths are added to the array EN. In the third module, if the threshold is not reached, other nodes should be reactivated. However, nodes will not be disabled, if the new performance is near the initial one (which is the case most of the time) because devices of the backup paths must stay active even if they are unneeded.

4.8.3 System evaluation

In order to prove the effectiveness of our power-aware system, this algorithm has been also realized using ns-3. Without loss of generality, we used Flatnet[33] and PTNet[72] data centers to evaluate the system since they are recent topologies that demonstrated a good performance compared to several well known data centers. Flatnet is composed of a total of n^3 servers (n is the number of ports per switch) and $2n^2$ external and internal switches. Each external switch connected to n servers is considered as a cluster. PTNet is composed of sn^2 servers (n is the number of ports per switch and s is a parameter to enlarge the network) and sn switches. The total number of ports in PTNet data center is equal to $(2sn(2n-1)+2n\sum_{i=1}^{s-1} i)$. In this simulation, we fix $s = n$. A cluster in PTNet includes one switch and its connected servers.
The simulation configuration is the same as the previous routing algorithm which is summarized in Table 3.9 in chapter 3.
Based on equation (4.10), the percentage of power saved in Flatnet is calculated as follows:

$$\%ES = \frac{\sum P_{disabled} * 100}{2n^2 * n + n^3 * 2} = \frac{\sum P_{disabled} * 100}{4 * n^3} \tag{4.14}$$

And, the percentage of power saved in PTNet is calculated as follows:

$$\%ES = \frac{\sum P_{disabled} * 100}{2sn(2n-1) + 2n\sum_{i=1}^{s-1} i} = \frac{\sum P_{disabled} * 100}{4n^3 - 2n^2 + 2n\sum_{i=1}^{n-1} i} \tag{4.15}$$

For a 10GbE equipment, the maximum power consumption of a switching port is 12 Watts [71]. Thus, we estimated the amount of energy saved by the proposed system based on the 12 watts maximum power consumption of the network equipment.
In our experiments, we used a Many-to-Many traffic pattern to evaluate the network in

a realistic environment. In fact, we have randomly created a traffic distribution where we have fixed a percentage of load (percentage of devices that send or receive packets). The combination of sources and destinations is chosen randomly.

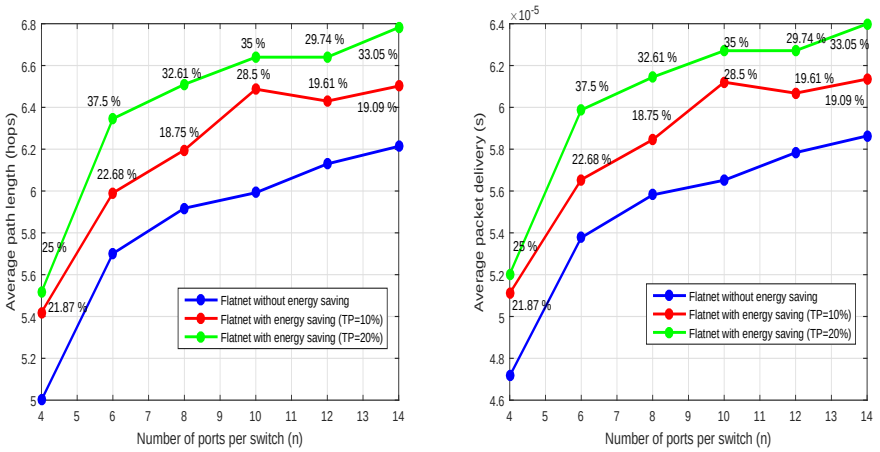
4.8.3.1 Trade-off between power saving and system performance

Figures 4.12 and 4.13 exhibit the power saving routing algorithm applied on Flatnet and PTNet topologies and show the performance of the network with different sizes by varying n. Indeed, for each n, at the same instant t, we randomly generate a traffic matrix and calculate the performance of the original routing and the power-aware routing in terms of average path length (Figure 4.12(a), Figure 4.13(a)), average packet delay (Figure 4.12(b), Figure 4.13(b)), throughput (Figure 4.12(c), Figure 4.13(c)) and saved energy (provided as percentages in the graphs). We suppose that the threshold percentage tolerated by the system is TP=10% for red graphs and TP=20% for the green graphs. We suppose also that the traffic load is 50%. For Algorithm 10, the configuration ($Ad_1 \cong Ad_0$) and ($Apl_1 \cong Apl_0$) means that (Apl_1, Ad_1) is close to the initial performance (Apl_0, Ad_0) by 5% if $TP > 10\%$ and 2% if $TP \leq 10\%$. We can see that with a concession of a maximum of 10% in the aforementioned performance metrics (TP=10%), we can achieve over 25% of energy saving for Flatnet and 20% for PTNet. And, with a concession up to 20% of the performance (TP=20%), we can save over 35% of energy for Flatnet and PTNet. A little more concession in term of performance leads to more energy saving. The graphs of APL and average packet delivery have similar tendencies because these two metrics are approximately linearly dependant. In fact, the APL is the number of hops used to reach the destination and the average packet delivery can be computed as follows:

$$Ad = \frac{1}{n_p} \sum_{i=1}^{n_p} Apl_i(d_i(S) + d_i(SW) + d_T) \tag{4.16}$$

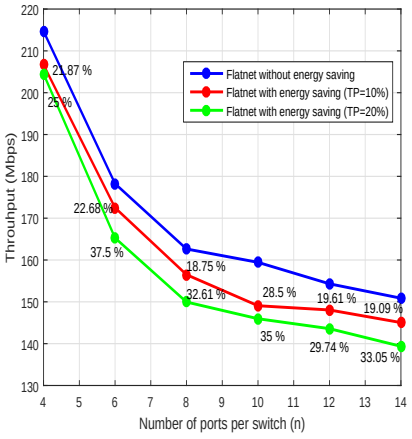
where Ad is the average packet delivery of the system, n_p is the total number of packets communicated in the system, Apl_i is the path length between two nodes when sending a packet i, $d_i(S)$ is the delay of processing in servers, $d_i(SW)$ is the delay of processing in switches and d_T denotes the transmission delay. Equation (4.16) shows that the Apl and Ad are dependent and have the same tendency as can be seen in Figures 4.12 and 4.13.

Figure 4.14 shows the energy consumed by Flatnet data center with its traditional routing algorithm and by applying the power-aware scheme. We can see that up to 3628 watts can be saved at a given time t when $n = 14$.



(a)

(b)

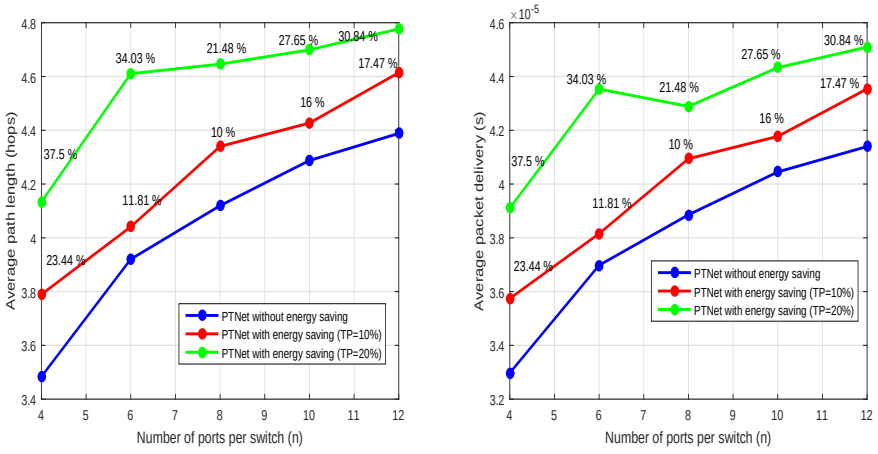


(c)

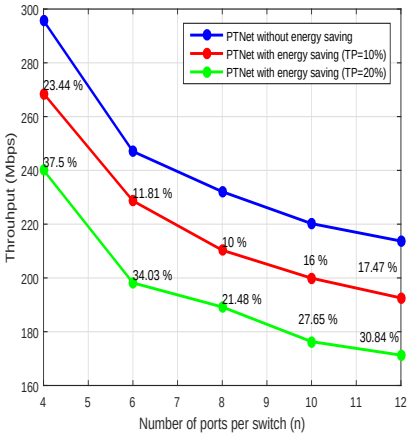
FIGURE 4.12: Trade-off between energy saving and system performance (Flatnet).

4.8.3.2 Trade-off between power saving and reliability

Figures 4.15 and 4.16 show the simulation results of the power saving under different fault tolerance levels. In PTNet and Flatnet, the maximum reliability level is 2 because the number of disjointed paths is equal to 2 (generally, the number of disjointed paths is equal to the number of ports per server). The performance threshold is fixed to 20% and the traffic load to 50%. Indeed, when turning off the maximum of idle devices for a better energy saving, the fault tolerance is not guaranteed. To ensure the robustness of the system, backup routes are added which means more nodes are active and more energy is wasted. The Figure 4.16 shows that up to 35% of energy can be saved with a reliability level equal to 1 and up to 20% for a reliability level equal to 2. More power saving can be achieved for lower fault tolerance. However, even by adding more backup devices, the power aware system still saves up to 3000 watts compared to the traditional system when $n = 14$. More energy can be saved for lower loads. The value



(a) (b)



(c)

FIGURE 4.13: Trade-off between energy saving and system performance (PTNet).

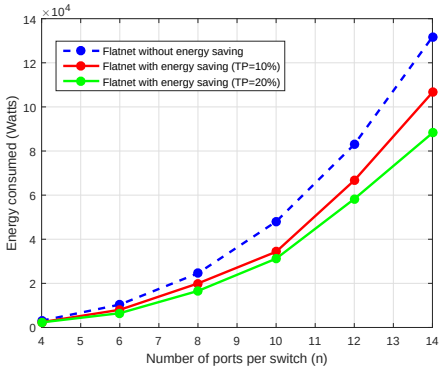


FIGURE 4.14: Energy saved (Watts).

of RL is decided by the administrator of the data center according to the requirement of the system in terms of reliability and power saving. From another perspective, the data center failure is low[76]. So after all, it is not wise to scarify so much energy for high level of reliability against small probability event.

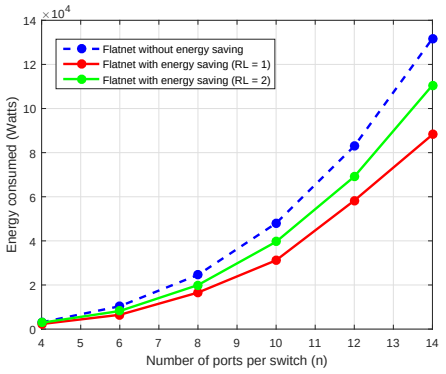


FIGURE 4.15: Reliability vs energy (Watts).

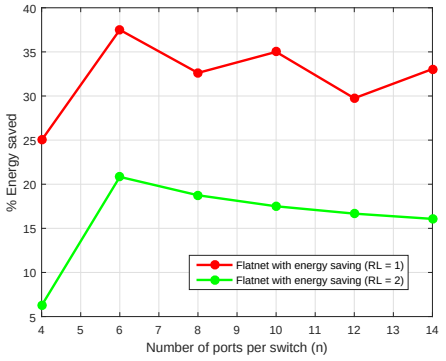


FIGURE 4.16: Reliability vs energy (%).

4.8.3.3 Different network loads

The network load is defined as the percentage of servers sending and receiving the data at a given time t. This factor has a great impact on the energy consumption. In fact, our system reaches its highest performance when applied to networks with reduced traffic loads, as in this case, the number of idle nodes is important. It means the number of vital nodes which should stay active is reduced. However, in data centers with high traffic, the number of idle nodes is small and therefore, there will be less energy saving. In this simulation, the TP is equal to 20% and RL is equal to 1. As shown in Figure 4.17, for different sizes of the network, when increasing the network load, the power conservation degrades. It means the system accomplishes more power saving at the lowest network load. When the load is equal to 100%, all nodes are included in the communication. So, no energy saving is possible. However, since the network utilisation is 5% to 25% most of the time, our power aware system saves from 30% to 50 % of energy when applied on Flatnet topology. A high energy saving (near 100%) is not possible for low loads because articulation nodes can not be powered off (without the articulation servers, the network will be disconnected).

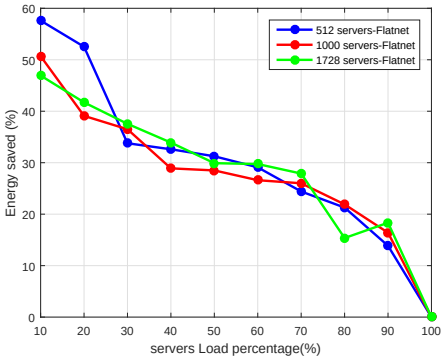
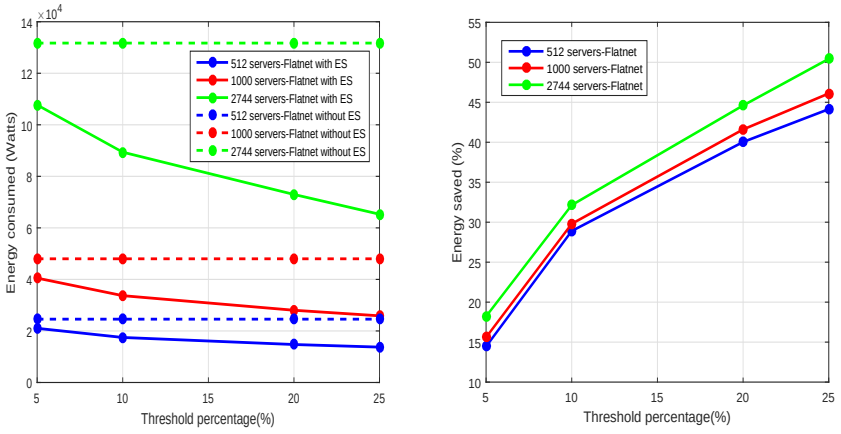


FIGURE 4.17: Load vs energy (%).

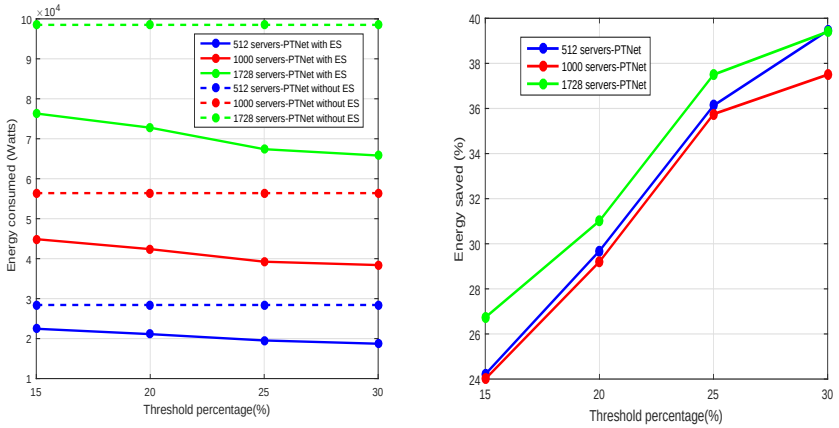
4.8.3.4 Different threshold percentages

The proposed power-aware routing algorithm achieves a trade off between saved energy and performance. Figures 4.18 and 4.19 show that for 50% load and first level reliability, we can achieve 15% to 50% of power saving and more than 5000 Watts can be saved at a given time t by using different thresholds. Moreover, by augmenting the threshold percentage, we can increase further the power saving. In fact, maintaining a better performance (decreasing the threshold) requires activating more servers and switches which offers more paths and shorter routes for communications to be delivered faster and without congestion. However, activating the minimum number of nodes obliges the packets to find alternative routes to avoid congestion which increases the average path length and the latency.



(a) (b)

FIGURE 4.18: Threshold vs energy (Flatnet).



(a) (b)

FIGURE 4.19: Threshold vs energy (PTNet).

4.8.3.5 Computation efficiency

The most important advantage of our approach is its computation efficiency. In fact, searching for nodes impacted in the communication traffic in real time requires high computational complexity and a lot of time. It is mainly because of the large searching space especially in huge data centers. However, in our proposed approach, the search of nodes to keep active is done in the initialisation of the system to avoid adding a burden of time to the routing algorithm. Then, at a given time t when receiving the traffic matrix, the receiver and sender nodes are kept active in addition to a subset of vital nodes chosen by order of importance from the pre-calculated table. The complexity of applying PCA to rank the nodes by importance in the array RN (in algorithm 10) is $O(p^2V + p^3)$ where p is the number of features which is 2 in our case and V is the data points which is the number of nodes. Thus, the complexity of PCA is $O(4V + 8)$. In addition the complexity of applying DFS is $O(V + E)$ where V is the number of nodes and E is the number of edges. The complexity of the traditional routing algorithm of the system used to deliver the flows of communication depends on the topology. For the case of Flatnet and PTNet, the routing tables are calculated only once and can then be used in real time with O(1)-complexity[33][72].

4.9 Further discussion

- Many contributions targeted reducing power consumption by using green routing algorithms approach. These approaches are applied on different types of networks. In general networks (especially the small ones), complexity of the algorithm is not always a constraint. However, when designing a data center, the

exponential growth of the network should be usually a restriction to consider. Hence, the first goal, in our work, is to propose a low complexity algorithm. In the first algorithm, we used the known architecture of PTNet to design green heuristic and maintain the connectivity and redundancy of the network. The regularity of the data center architectures is a strong enabler to build a green network. In the second algorithm, calculating the vital nodes during the built of the network was a way to simplify the searching of idle nodes. This reduced searching time is a competitive edge of our work compared to many contributions owning a high calculation complexity.

- In our second algorithm, we calculated the criticality of the nodes based on multiple parameters. Several works have studied the aspect of critical nodes such as searching the important nodes based on betweeness[91] or closeness[92], etc. These indexes either consider the topology of the network (number of shortest paths, connectivity of a node, distance between different nodes) or the load passing by different network devices. However, in our work, we take into consideration all aspects of the network by calculating a single index deduced from different parameters. Also, the traffic load is considered and the connectivity of the network is always the most important criterion to respect. Additionally, the vital nodes are used in our contribution for power saving purposes. Some references calculate the critical nodes to consolidate the traffic on a subset of network elements. For examples, authors in [93] proposed two indexes: G-game and L-game to choose important nodes and links in a general wired network. These indexes take into consideration the traffic load, the redundancy and the volume of traffic transported by nodes. However, calculating the proposed indexes is computationally intensive when considering a realistic and complex network scenarios. In our case, to calculate vital nodes, the network is divided into clusters and therefore, the complexity is minimized. Also, this task is done during the built of the network to reduce the searching time. Moreover, performance thresholds are chosen by the administrator to ensure the network QoS.

- In this chapter, we proposed two power aware routing algorithms enjoying the characteristics summarized in Table 4.4. The first algorithm is designed for PTNet. It uses the already-known interconnection of this topology to quickly find the minimum active subset and keep the network non-partitioned. It means, the computation of active nodes has a very low complexity and a negligible time. If a data center has a known and regular architecture, this approach can be applied. If not, the routing algorithm based on vital nodes is adequate for any data center network with a low computation complexity (even if it is higher than the first one) and a very efficient power saving.

TABLE 4.4: Comparison between the power aware algorithms.

Prameters	Power aware routing algorithm for PTNet	Power aware routing algorithm based on vital nodes
Computation complexity	Very low	Low
Computation time	Very low	Low
Power efficiency	Meduim	High
Reliability	2 disjointed paths (Maximum for PTNet)	Configured $\geqslant 1$
Performance of the network	Threshold chosen by the administrator	Threshold chosen by the administrator
Compatibility with DCs	Designed for PTNet	All DCs

- The objective of this work was to reduce the energy consumed by idle nodes in order to establish the proportionality between the traffic load and the power consumption. In this study, the energy consumed by the cooling machines, chasis and linecards is not considered. It is certainly interesting to investigate the possibility of building a proportionality between the workload and the energy consumed by the fans and cooling machines. Fan controllers can be added to the data center machines to tune the fan speed dynamically and minimize the cooling power according to the workload.

4.10 Conclusion

In this chapter, we address the problem of non proportionality between the traffic load and the power consumption from a routing perspective. We first introduced an energy optimization scheme which is used to establish the performance guaranteed power-aware systems. Then, we proposed two schemes where we keep active only the minimum number of network devices. In addition, we made a reasonable trade-off between the performance, the reliability and the power consumption by applying different heuristics. These approaches grant a maximum power saving reaching up to 40% in low loads and more than 30% in high loads. The new systems provide also a reasonable computation time with a little concession in performance tolerated by the system administrator. The experiments, conducted under different conditions (different loads,

different thresholds, different network scales and different reliability levels), confirm the efficiency of the proposed power aware routing algorithms.

Chapter 5

Greening data center networks: Vacation Queuing model

5.1 Introduction

Our Studies conducted in previous chapters stated that the network operates only at 5% to 25% of its maximum capacity and the traffic loads vary depending on the period which means the utilization of the network fluctuates according to these loads. In this context, we proved that the traffic can be satisfied by only a subset of the network and by powering off the idle servers, the power consumption becomes proportional to the traffic load. However, defining the nodes to power off depends on the traffic matrix which is an unpredictable metric and needs a high computation time and complex calculations. Furthermore, energy conservation is insignificant in high loads. This is explained by the fact that incoming packets are stochastic and can be dense in certain periods and sparse in others and nodes have to be kept powered on, computing and waiting for arriving jobs.

To build an energy efficient data center independent from the traffic load, we should consider two facts: First, the redundancy, ensuring a bandwidth enhancement and load balancing, contributes the most in wasting a large amount of energy. However, since the maximum capacity of the network is not reached in most of the time, redundancy can be set as optional. Second, instead of activating all ports of devices to wait for the randomly arriving tasks, packets incoming from different links can be relayed to a defined active ports while forcing the other ones to be in a low power mode.

To implement this idea, a re-architecturing of network devices is proposed and a vacation queuing model is applied to analyze the power consumption in the proposed

system. The main contributions in this work can be summarized as follows: (i) Proposing a re-architecturing of the network devices (ii) Presenting a packet scheduling algorithm that manages the distribution of packets among different ports. (iii) Analyzing the proposed approach by applying the vacation queuing theory (iv) Evaluating the trade-off between the waiting time of packets in the queue, the throughput and the energy saved.

5.2 Motivation

Most of the proposed solutions to green the data centers are working on receiving the traffic load, calculating the nodes implicated in the communication matrix and putting the idle interfaces into sleep or lower rate status. However, this kind of efforts faces a fundamental problem which is the unpredictability of the incoming traffic. Also, because of the burst nature of the traffic, energy saving is important only in low loads and there is almost no saving for high loads. In addition, since the traffic can be satisfied by only a subset of devices and the idle ones can be powered off, the fault tolerance and the throughput can be affected. To overcome this challenge, we propose a re-architecture of network devices and a Vacation/Service algorithm to use resources more efficiently and minimize the dependency on the traffic load without affecting the fault tolerance or adding a computation time.

5.3 Proposed approach

5.3.1 Re-architecturing the network devices

Figure 5.1 shows a typical hardware architecture of a network device vs the proposed re-architectured device. Unlike the conventional network device (switch, server, router,...), presented in Figure 5.1(a), where every interface has its own processing unit to run routing protocols and decide how to forward packets, routing decisions are stripped from interfaces. As a result, in our approach presented in Figure 5.1(b), the interfaces level is simply responsible for receiving, gathering, and forwarding packets to the controller level. The controller level decides what is the available unit to process the packets. The processing unit level is responsible to process, extract and decode the packet header, look up to the destination address and decide what to do with it (forward or drop)[94]. If the queue of one unit is congested, it notifies the controller to forward the incoming packets to the next unit (see Figure 5.2). The processed packets pass by the controller level again to be relayed to the appropriate interface.

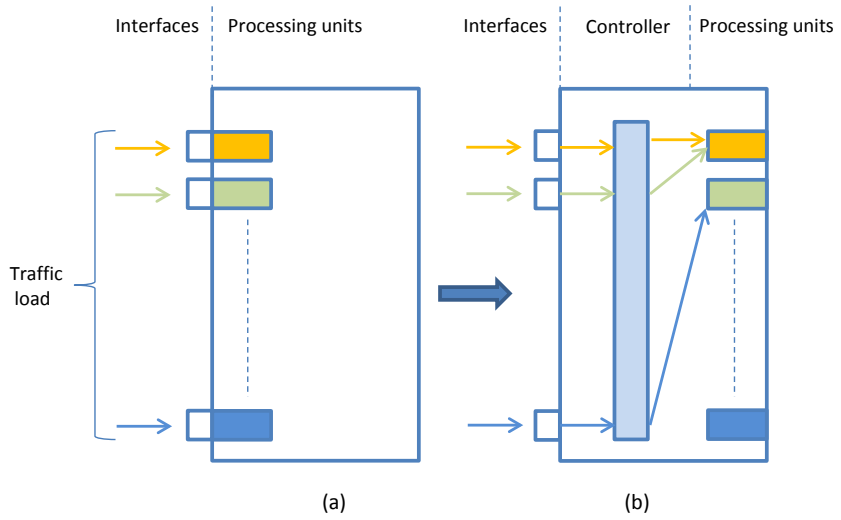


FIGURE 5.1: Re-architecturing the network devices.

The objective of the re-architecturing is that the incoming packets can be handled by any one of the processing units if it is available while respecting the Vacation/Service algorithm described in the section 5.3.2. Thus, in low loads, instead of using all units, the traffic incoming/directed to n ports can be handled by a lower number of processors (n is the number of ports per device). Then, the idle ones can be turned into sleep status which saves a considerable amount of energy, reduces the dependency on the unpredictable traffic and removes any computation complexity.
This new architecture, however, requires new hardware design. In fact, a similar network hardware has been proposed and implemented for the Software-Defined Networking (SDN) [95] for another purpose. In SDN, the network devices are not responsible any more for the routing decisions. A new layer is added where new switches/servers have the role of processing the data (processing units in our case). As SDN becomes a trend for cloud computing, the industry is paying more attention to this decoupled network hardware including NetFPGA[96]. Therefore, the proposed re-architecturing can be available as another alternative.

5.3.2 Vacation/Service algorithm

Given the proposed re-architecturing described above, it now becomes possible to merge packets from multiple input/output interfaces to be processed by few units. However, an algorithm to manage the distribution of tasks between different processing units and the vacation times attributed to each unit is important to maximize the sleeping units and, hence, minimize the energy waste.
In the initialization of the system, all processing units are deactivated. After a vacation

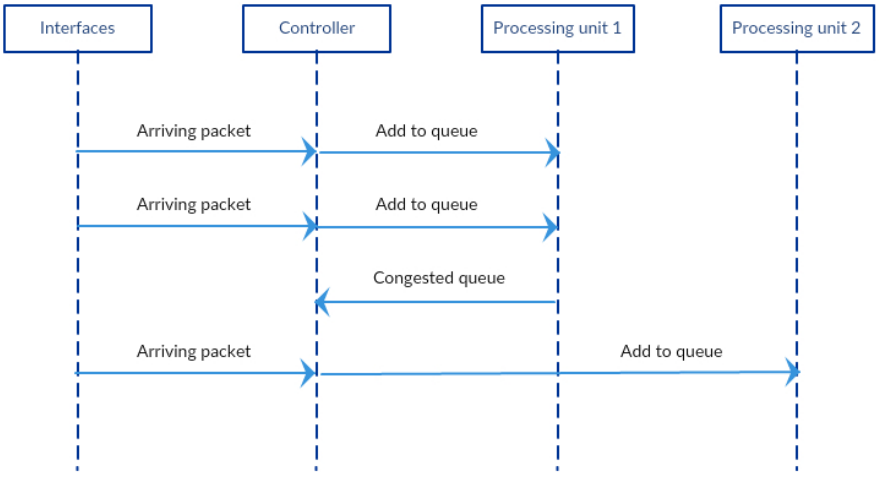


FIGURE 5.2: Interaction between components of the re-architectured device.

time V_1 initially equal to T_{min}, their buffers are examined. If there are waiting packets, the related unit will be activated, otherwise it will initiate another vacation time V_2. The first coming packets are routed automatically by the controller to the first processing unit. When the buffer size is equal to K, the queue size (the unit is congested), the unit notifies the controller and the arriving packets are routed to the adjacent unit.
From the controller perspective, packets are sent to the processor units by order. If the first one is congested (buffer is equal to K), the packets are routed to the next available one.
From the processors perspective, each unit can experience five states (vacation, listening, wake-up and service). The transition between the vacation and the service state depends on the result of the listening time T_l. In fact, when the vacation time elapses, the unit switches to listening where it examines the buffer. If there are awaiting packets, the unit must start the service period where it will be active. However, to start serving the jobs, it passes first by the wake-up period T_w where it warms-up between sleeping and active state. If the listening does not trigger any awaiting packet, the processor unit starts another vacation time. The listening is performed after every vacation time under a low energy rate. Figure 5.3 describes the passage between the different states experienced by the processing unit.

5.3.3 Vacation Queuing model for a single processing unit

In this section, we will analyse our system from a queuing perspective. First, we will begin to study the model for one unit and then we will establish the relation between units in section 5.3.4. So far, we will consider an $M/G/1/K$ queue (**M**: arrivals are Markovian, **G**: service have general times distribution, **1**: queue for only one unit, **K**:

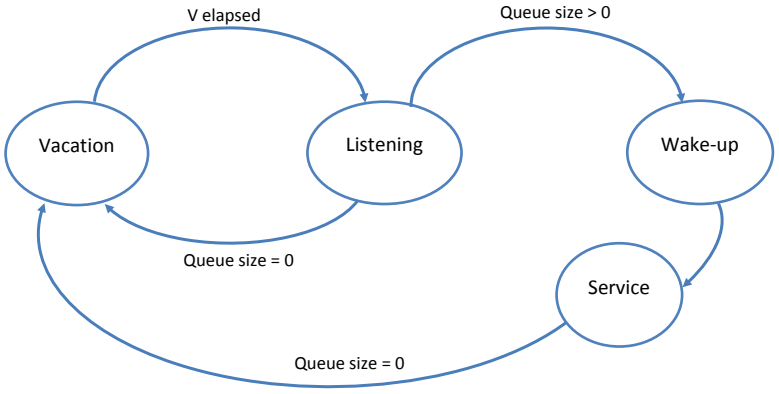


FIGURE 5.3: Processing unit state diagram.

maximum number of packets in the queue) in which the processing units go on vacation for predefined periods if the the queue size is equal to 0 [61]. Packets are assumed to arrive according to an independent Poisson process [97] with a data rate equal to λ. The data rate is the number of packets received in a time unit (Tu). The service times are generally distributed with pdf $s(t)$, cdf $S(t)$. The service time is the time needed to process a single packet. Let the mean service time be equal to $E[\sigma]$. We assume also that the vacation times have a pdf equal to $v(t)$, a cdf equal to $V(t)$ and a Laplace Stieltjes transform equal to $L_V(s)$. Let the mean vacation times be equal to $\bar{v}$.

Note that the queue size impacts the duration of the service period and it is itself impacted by the length of the last vacation time. When the queue empties, a new cycle is initialized. Each cycle consists of:

- A vacation period: Let V denote the vacation period where the unit is sleeping. It is composed of N vacation times denoted as $V_1, V_2, ..., V_N$. Each vacation time ends with a listening time equal to T_l.

- A wakeup period: Its duration is fixed and depends on the equipment used in the data center. It is denoted by T_w. In this period, the machine warms up to start processing the awaiting packets.

- A service period: Let S denote the period where the unit is active. During this period, new customers may arrive in the system. Let k denote the number of jobs waiting to be processed in the unit queue. Jobs will be processed in slots of time called service times $s_1, s_2, ..., s_k$. Service times are the sub-periods of the service period. Each job should wait for the previous job to be processed to initiate its service time.

Figure 5.4 and Table 5.1 show the notations introduced so far. In Figure 5.4, the first packet appears in the fourth vacation time V_4 and packets keep arriving while the unit is processing the awaiting jobs. In the sixth service time, the last packet is being processed and a vacation period will start since the queue becomes empty. The introduction of some notations in the Table 5.1 will be deferred to ulterior sections.

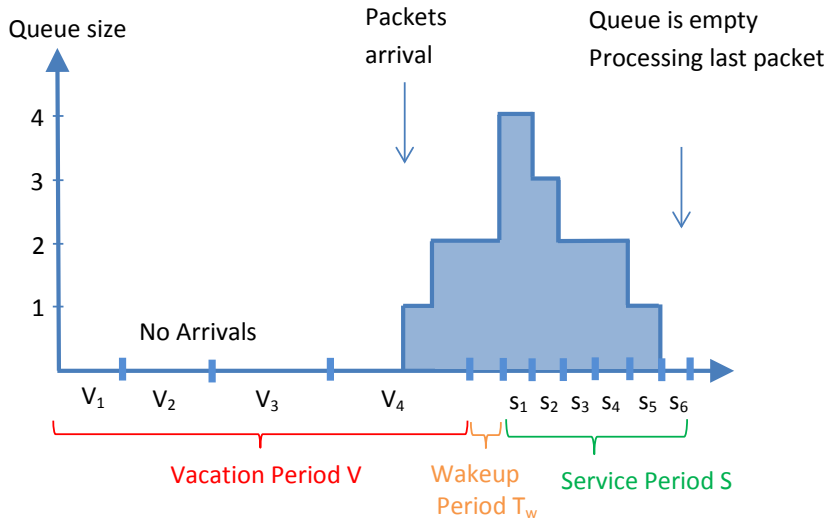


FIGURE 5.4: The queue size and corresponding status.

5.3.3.1 Expectation of waiting time and size of the queue

For our study of the $M/G/1/K$ queue with *exhaustive service* and *multiple vacations*, we will use an imbedded Markov Chain approach in [98][99]. The performance measures that can be calculated using this approach are the mean response time $E[R]$ of a packet which is the mean time calculated from the arrival in the system to packet process completion and the mean waiting time in the queue which is defined as:

$$E[W] = E[R] - E[\sigma] \tag{5.1}$$

In addition, we will calculate the blocking probability P_B which is the probability to relay the packet to the next unit if the queue is full ($(1 - P_B)$ is the probability that an arrived packet is accepted) and the effective data rate of the system which is the mean number of packets that are actually served by the unit and is given by:

$$\lambda_U = \lambda(1 - P_B) \tag{5.2}$$

We also introduce the offered load which is defined by Little's law[61] as:

$$\rho = \lambda E[\sigma] \tag{5.3}$$

Since some of the arrivals will be blocked and will be forwarded to the queue of the next unit, the effective carried load will be defined as:

$$\rho_U = \rho(1 - P_B) \tag{5.4}$$

TABLE 5.1: Summary of notations.

Notation	Description
K	Maximum queue size
λ	Data rate of Poisson process
S	Service period
$s(t)$	pdf of service distribution function
$S(t)$	cdf of service distribution function
$E[\sigma]$	mean service time
$E[S]$	mean service period
V	Vacation period
$v(t)$	pdf of vacation distribution function
$V(t)$	cdf of vacation distribution function
$L_V(s)$	Laplace Stieltjes transform of the vacation times
$E[V]$	Mean vacation period
$\bar{v}$	Mean vacation time
T_l	Listening period
T_w	Wake-up period
$E[R]$	Mean response time
$E[W]$	Mean waiting time
P_B	Blocking probability
T_{min}	Size of the first vacation time
ρ	Offered load
ρ_U	Effective carried load
λ_U	Effective data rate
$E[L]$	Mean queue size
q_k	Probability of k jobs waiting when the vacation time ends
Q_k	Probability of k jobs waiting during the vacation period
π_k	Probability of k jobs waiting when the service time ends
Π_k	probability of k jobs waiting during the service period
f_j	Probability of j arrivals after a vacation time
F_j	Probability of j arrivals during vacation period
a_j	Probability of j arrivals in the service time
A_j	probability of j arrivals in the service period
η	Mean time between two Markov points
$E[N]$	Mean number of vacation times
n	Number of ports per networking device

The blocking probability can be written then as:

$$P_B = \frac{\rho - \rho_U}{\rho} \tag{5.5}$$

Another performance measure that we are going to consider is the mean number of jobs present in the queue at a random time noted $E[L]$. By applying the Little's law, we can have the relation:

$$E[L] = \lambda_U E[R] \tag{5.6}$$

To evaluate these measures, the Markov imbedded points are chosen from time instants when a vacation time ends or a service time ends. We supposed here that the wake-up period is small and is not enough to receive packets. Figure 5.5 illustrates the imbedded points described in this section and marked with red circles.

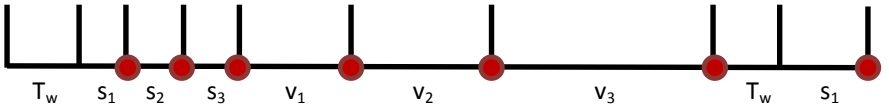


FIGURE 5.5: Imbedded Markov points for *M/G/1/K* queue.

We will consider:
q_k: probability of k jobs waiting when the vacation time ends (k=0,1,...,K).
π_k: probability of k jobs waiting when the service time ends (k=0,1,...,K-1).
Note that after a service time, the queue size cannot be K because at least one packet was treated.
f_j: probability of j arrivals during a vacation time (j=0,1,...,∞).
a_j: probability of j arrivals during a service time (j=0,1,...,∞).
Since arrivals are assumed to form a Poisson process:

$$f_j = \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t} v(t) dt, \tag{5.7}$$

where $v(t)$ is the probability density function of the vacation times.

$$a_j = \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t} s(t) dt, \tag{5.8}$$

where $s(t)$ is the probability density function of the service times.
After an imbedded point, the system state can be written as follows:

$$q_k = (q_0 + \pi_0) f_k \qquad k = 0, 1, ..., (K-1) \tag{5.9}$$

$$q_K = (q_0 + \pi_0) \sum_{k=K}^{\infty} f_k \qquad k = K \tag{5.10}$$

$$\pi_k = \sum_{j=1}^{k+1} (q_j + \pi_j) a_{k-j+1} \qquad k = 0, 1, ..., (K-2) \tag{5.11}$$

$$\pi_{K-1} = q_K + \sum_{j=1}^{K-1} (q_j + \pi_j) \sum_{k=K-j}^{\infty} a_k \qquad k = K-1 \tag{5.12}$$

Since the sum of all probabilities is equal to 1, we will get:

$$\sum_{k=0}^{K} q_k + \sum_{k=0}^{K-1} \pi_k = 1 \tag{5.13}$$

The probability that an arbitrary Markov point will be followed by a vacation time is $(q_0 + \pi_0)$. In other word, $(1 - q_0 - \pi_0)$ is the probability that a service time will follow the arbitrary Markov point. Thus, the mean time η between successive Markov points is given by:

$$\eta = (q_0 + \pi_0)\bar{v} + (1 - q_0 - \pi_0)E[\sigma] \tag{5.14}$$

The effective carried load is also defined as the probability that the server is busy at an arbitrary time. Therefore, it can be defined as:

$$\rho_U = \frac{(1 - q_0 - \pi_0)E[\sigma]}{(q_0 + \pi_0)\bar{v} + (1 - q_0 - \pi_0)E[\sigma]} = \frac{(1 - q_0 - \pi_0)E[\sigma]}{\eta} \tag{5.15}$$

From (5.14) and (5.15), we can write:

$$q_0 + \pi_0 = \frac{\eta(1 - \rho_U)}{\bar{v}} \tag{5.16}$$

Until this step, we analyzed the queue size distribution at points corresponding to service time completion or vacation time completion. However, to study the system more deeply, we need to analyze the distribution at an arbitrary point (during service and vacation times) as then we can find the mean number of packets in the queue, the mean waiting time, the mean response time and the other related parameters. Hence, we will consider:

Q_k: probability of k jobs waiting during the vacation period (k=0,1,...,K).

Π_k: probability of k jobs waiting during the service period (k=0,1,...,K-1).

F_j: probability of j arrivals during the vacation period (j=0,1,...,∞).

A_j: probability of j arrivals during the service period (j=0,1,...,∞).

F_j can be evaluated using equation (5.7):

$$F_j = \sum_{i=j}^{\infty} f_i = \sum_{i=j}^{\infty} \int_0^{\infty} \frac{(\lambda t)^i}{i!} e^{-\lambda t} v(t) dt \tag{5.17}$$

We can prove the following expression (see appendix A):

$$F_j = \lambda \int_0^{\infty} \frac{(\lambda t)^{(j-1)}}{(j-1)!} e^{-\lambda t} [1 - V(t)] dt \tag{5.18}$$

And

$$\sum_{j=1}^{\infty} F_j = \lambda \bar{v} \tag{5.19}$$

Similarly we can calculate A_j (see appendix A):

$$\begin{aligned} A_j &= \sum_{i=j}^{\infty} a_i = \int_0^{\infty} \frac{(\lambda t)^j}{j!} e^{-\lambda t} s(t) dt \\ &= \lambda \int_0^{\infty} \frac{(\lambda t)^{(j-1)}}{(j-1)!} e^{-\lambda t} [1 - S(t)] dt \end{aligned} \tag{5.20}$$

and

$$\sum_{j=1}^{\infty} A_j = \lambda E[\sigma] = \rho \tag{5.21}$$

In order to calculate Q_k, we will suppose that we have an arbitrary point occurring in the vacation period where there have been k arrivals since the start of this period. Let x denote the period between the start of the vacation time and the selected instant. Note that the probability of selecting a vacation interval is $(1 - \rho_U)$ (since the probability of selecting a service time is ρ_U, see equation (5.15)). Note also that the pdf of the interval x is given by the residual time $\frac{1 - V(x)}{\bar{v}}$[61]. We can write:

$$\begin{aligned} Q_k &= (1 - \rho_U) \int_0^{\infty} \frac{(\lambda x)^k}{k!} e^{-\lambda x} \frac{1 - V(x)}{\bar{v}} dx && k = 0, ..., (K-1) \\ &= (1 - \rho_U) \sum_{k=K}^{\infty} \int_0^{\infty} \frac{(\lambda x)^k}{k!} e^{-\lambda x} \frac{1 - V(x)}{\bar{v}} dx && k = K \end{aligned} \tag{5.22}$$

The equation (5.22) can be simplified using equation (5.18):

$$\begin{aligned} Q_k &= \frac{(1 - \rho_U)}{\lambda \bar{v}} F_{k+1} && k = 0, ..., (K-1) \\ &= \frac{(1 - \rho_U)}{\lambda \bar{v}} \sum_{k=K+1}^{\infty} F_k && k = K \end{aligned} \tag{5.23}$$

From equation (5.9), we can show that :

$$\sum_{j=k}^{K} q_j = (q_0 + \pi_0) \sum_{j=k}^{k} f_j = (q_0 + \pi_0) F_k \qquad k = 0, ..., K \tag{5.24}$$

Using equations (5.24) and (5.16), and knowing that $\sum_{j=1}^{\infty} F_j = \sum_{j=1}^{\infty} j f_j$ [98], we can write:

$$\begin{aligned} Q_k &= \frac{(1 - \rho_U)}{\lambda \bar{v} (q_0 + \pi_0)} \sum_{j=k+1}^{K} q_j = \frac{1}{\lambda \eta} \sum_{j=k+1}^{K} q_j && k = 0, ..., (K-1) \\ &= 1 - \rho_U - \frac{(1 - \rho_U)}{\lambda \bar{v} (q_0 + \pi_0)} \sum_{j=1}^{K} j q_j = 1 - \rho_U - \frac{1}{\lambda \eta} \sum_{j=1}^{K} j q_j && k = K \end{aligned} \tag{5.25}$$

To calculate Π_k, we will similarly suppose that we have an arbitrary point occurring in the service period where there have been k arrivals since the start of this period. Let x denote the period between the start of the service time and the selected instant. Note that the probability of selecting a service time is ρ_U. Let's suppose that the service time

begins with j waiting packets. Thus, the probability of having j jobs will be:

$$\begin{cases} q_j + \pi_j & j = 1, ..., K-1 \\ q_K & j = K \end{cases} \tag{5.26}$$

Note also that the pdf of the interval x is given by the residual time $\frac{1-S(x)}{E[\sigma]}$. We can write:

$$\begin{aligned} \Pi_k &= \rho_U \textstyle\sum_{j=1}^{k}(q_j + \pi_j) \int_0^\infty \frac{(\lambda x)^{k-j}}{(k-j)!} e^{-\lambda x} \frac{1-S(x)}{E[\sigma]} dx \qquad k = 0, ..., (K-1) \\ &= \rho_U q_K + \rho_U \textstyle\sum_{j=1}^{k-2}(q_j + \pi_j) \sum_{k=K-j}^{\infty} \int_0^\infty \frac{(\lambda x)^{k}}{(k)!} e^{-\lambda x} \frac{1-S(x)}{E[\sigma]} dx \qquad k = K \end{aligned} \tag{5.27}$$

Similar to Q_k, we can get the final expression of Π_k as follows:

$$\begin{aligned} \Pi_k &= \frac{1}{\lambda\eta}(\pi_k - \textstyle\sum_{j=k+1}^{K} q_j) \qquad\qquad k = 0, ..., (K-1) \\ &= \frac{\rho_U(\rho-1)}{\rho} + \frac{1}{\lambda\eta}\textstyle\sum_{j=1}^{K} j q_j \qquad\qquad k = K \end{aligned} \tag{5.28}$$

We define L as the size of the queue at a random time. This probability can be deduced from the values of Q_k and Π_k:

$$\begin{aligned} &Prob[L=0] = Q_0 \\ &Prob[L=k] = Q_k + \Pi_k = \frac{\pi_k}{\lambda\eta} \qquad\qquad k = 0, ..., (K-1) \\ &Prob[L=K] = Q_K + \Pi_K = \frac{(\rho - \rho_U)}{\rho} \end{aligned} \tag{5.29}$$

Thus, the mean number of jobs present in the queue at a random instant is given by:

$$E[L] = \textstyle\sum_{k=0}^{K} k Prob[L=k] = K\frac{(\rho-\rho_U)}{\rho} + \sum_{k=1}^{K-1} k\frac{\pi_k}{\lambda\eta} \tag{5.30}$$

We can now derive the response time from equation (5.6). Since each packet can wait at most for one wake-up period T_w, it will be added to the mean response time.

$$E[R] = \frac{E[L]}{\lambda_U} + T_w \tag{5.31}$$

The mean waiting time in equation (5.1) can also be expressed as follows:

$$E[W] = \frac{E[L]}{\lambda_U} + T_w - E[\sigma] \tag{5.32}$$

5.3.3.2 Expectation of vacation period

We noted that V presents the vacation period (where the processing unit is sleeping). The period V consists of N vacation times denoted as $V_1, ..., V_N$. V_N is the last vacation time which means that the listening process reported the arrival of packets. To compute the mean number of vacation times in the vacation period V, we will use the method in [100]. Let $\widehat{V}_1, ..., \widehat{V}_N$ refer to the instants when the periods $V_1, ..., V_N$ ends. We can see that the vacation period ends at $\widehat{V}_N$ and $V = \widehat{V}_N = \sum_{j=1}^{N} V_j$. We note also that when $N \geq i$, it means that there is no arrival of packets during $\widehat{V}_{i-1} = \sum_{j=1}^{i-1} V_j$. Let EV_i represent this event (no arrival during $\widehat{V}_i$) and EV_i^c denote the complementary event. The Laplace Stieltjes transform of V_i can be written as follows:

$$L_{V_i}(\lambda) = E[e^{-\lambda V_i}]$$

We can calculate the probability of having a certain number of vacation times as follows:

$$\begin{aligned} P(N = 1) &= P(EV_1^c) \\ &= 1 - P(EV_1) \\ &= E[1 - e^{-\lambda V_1}] \\ &= 1 - L_{V_1}(\lambda) \end{aligned}$$

$$\begin{aligned} P(N \geq i) &= \prod_{j=1}^{i-1} P(EV_j) \\ &= \prod_{j=1}^{i-1} L_{V_j}(\lambda) \end{aligned} \tag{5.33}$$

Using equation (5.33), the expectation of the number of vacation times can be written as:

$$\begin{aligned} E[N] &= \sum_{i=0}^{\infty} iP(N = i) \\ &= \sum_{i=0}^{\infty} P(N \geq i) \\ &= \sum_{i=0}^{\infty} \prod_{j=1}^{i-1} L_{V_j}(\lambda) \end{aligned} \tag{5.34}$$

As defined previously, the vacation period is expressed as:

$$\begin{aligned} V &= \sum_{i=1}^{N} V_i \\ &= \sum_{i=1}^{\infty} V_i \mathbb{1}\{N \geq i\}, \end{aligned}$$

Where, $\mathbb{1}\{N \geq i\}$ is equal to 1 when $N \geq i$ and 0 when $N < i$. We assume that the vacation times are independent and depend only on no arrival of packets during the listening period. The expectation of the vacation period is deduced using equation

(5.33):

$$E[V] = \sum_{i=1}^{\infty} E[V_i] \prod_{j=1}^{i-1} L_{Vj}(\lambda) \tag{5.35}$$

Hence, the mean vacation time can be written as follows:

$$\bar{v} = \frac{\sum_{i=1}^{N} E[V_i]}{E[N]} = \frac{E[V]}{E[N]} \tag{5.36}$$

5.3.3.3 Expectation of service period

To calculate the mean service period S where the unit is active, we divided it into k sub-periods $s_1 = ... = s_k$. These sub-periods are supposed to be independent and identically distributed. Therefore,

$$\begin{aligned} E[S] &= E[E[S|L]] \\ &= E[LE[s_k]] \\ &= E[L]E[s_k] \end{aligned}$$

Each sub-period can be considered as a simple $M/G/1/K$ queue without vacations where $\rho = \lambda E[\sigma] = \dfrac{E[s_k]}{E[s_k] + \dfrac{1}{\lambda}}$[61] (the fraction of time the port is working is equal to the mean service period divided by the mean cycle length). It means $E[s_k] = \dfrac{E[\sigma]}{1-\rho}$. Hence:

$$E[S] = E[L]\frac{E[\sigma]}{1-\rho} \tag{5.37}$$

5.3.3.4 Expectation of energy gain

Since a processing unit can experience different states including vacation, service, listening and wake-up, we can distinguish between three possible energy levels which are from the highest to the lowest: C_{high}, consumed during the processing of packets (service period); C_{listen}, experienced when checking the state of the queue (listening period) or in the wake-up period and C_{sleep}, consumed when the unit is inactive (vacation period). During sleeping period, we observe that there is $E[N]$ listening periods and one wake-up period T_w in every cycle. The energy consumption of the system can be written:

$$E_{powe-aware} = \frac{E[S]C_{high}}{(E[S] + T_w + E[V])} + \frac{(T_l E[N] + T_w)C_{listen}}{(E[S] + T_w + E[V])} + \frac{(E[V] - T_l E[N])C_{sleep}}{(E[S] + T_w + E[V])} \tag{5.38}$$

To calculate the energy gain, we will compare the proposed system described in Figure 5.6 to the always-on device presented in Figure 5.7. The power consumption of the always-on system is equal to C_{high} when it is active and equal to C_{low} in the idle state in case of no traffic load. The always-on system can experience also the rejection mechanism if the queue is congested (queue size equal to K). The blocking probability in the always-on system is given by Erlang C[101]:

$$P_B = \frac{\rho^K}{n^{K-n}n!}p_0, \tag{5.39}$$

where p_0 is equal to:

$$p_0 = \left\{ \sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n[1-(\frac{\rho}{n})^{K-n+1}]}{n!(1-\frac{\rho}{n})} \right\}^{-1}$$

The energy consumption of the always-on system can be calculated as follows:

$$E_{always-on} = \rho(1-P_B)C_{high} + (1-\rho(1-P_B))C_{low} \tag{5.40}$$

The economy of energy when using the power saving mechanism comparing to the original mechanism is equal to $E_{always-on} - E_{power-aware}$. Thus, we can define the relative energy gain as:

$$EG = \frac{E_{always-on} - E_{power-aware}}{E_{always-on}} \tag{5.41}$$

In practice, the energy consumed in sleep state is neglected compared to the energy consumed in the service state which means $C_{high} \gg C_{sleep}$ and $\frac{C_{sleep}}{C_{high}} \cong 0$. $E_{power-aware}$ can be reduced to:

$$E_{power-aware} = \frac{E[S]C_{high}}{(E[S]+T_w+E[V])} + \frac{(T_l E[N]+T_w)C_{listen}}{(E[S]+T_w+E[V])} \tag{5.42}$$

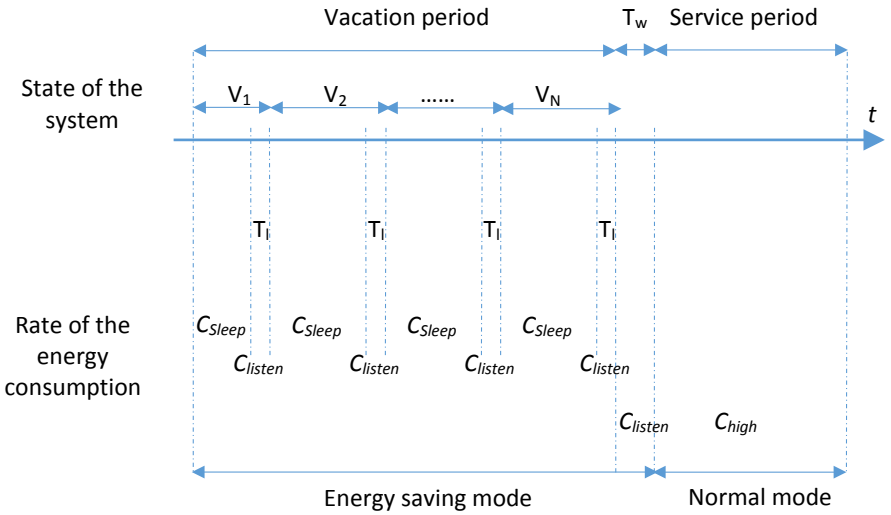


FIGURE 5.6: Power-aware system.

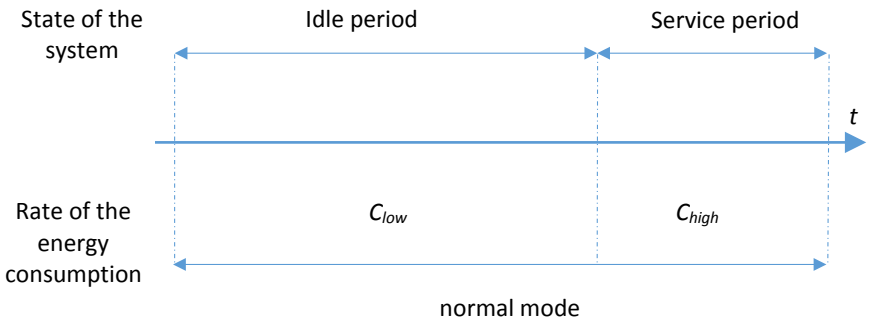


FIGURE 5.7: Always-on system.

5.3.4 The distribution of data rate among different processing units

Until now, we analyzed the system $M/G/1/K$ for a single unit queue with a Poisson arrival process. However, a network device has multiple processing units. Therefore, we need to generalize our study to an $M/G/n/K$ system (n is the number of units per network device). As presented in Figure 5.8, at a random time t, the unit can process only one packet and its queue has $(K-1)$ waiting positions where the jobs can wait if they find the unit busy on arrival. This queue can have K waiting positions if the unit is on vacation. Packets arriving when the system is full are not allowed to enter the queue and will be relayed to the next unit. In other word, the first unit rejects the blocked packets and sends them to the queue of the next unit.

Only a fraction $(1-P_B)$ of the arrivals actually enters the queue of the first unit (equation (5.5)). The effective arrival rate of packets waiting in the queue is only $\lambda_U = \lambda(1-P_B)$. Each unit experiences the rejection mechanism and relays the data to the

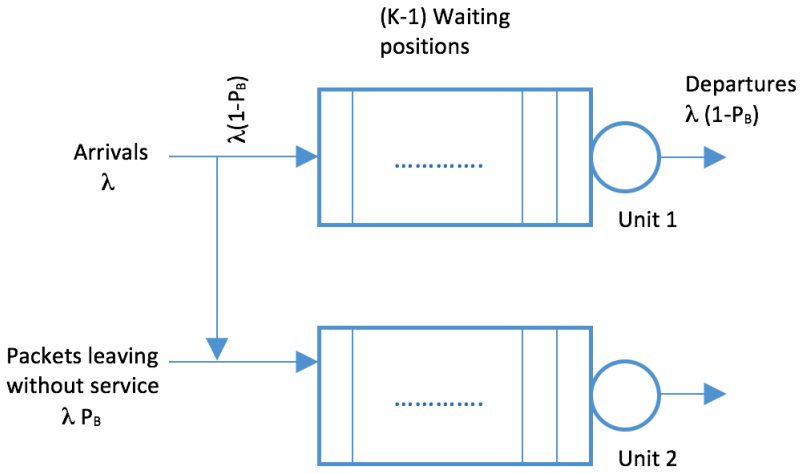


FIGURE 5.8: Distribution of effective data rate between units.

next unit. Therefore, the traffic load is distributed among units with different rates in such a way $\lambda_{U1} \geq \lambda_{U2} \geq ... \geq \lambda_{Un}$.

In this work, we assume that the blocked load leaving the first unit and entering the next unit follows a Poisson process with a parameter $\lambda_{U2} = \lambda P_B$. In the future, we can study exact distributions in different units and describe the new performance parameters (waiting time, queue length) and exact energy gain. Following this assumption and using equation (5.5), we can write the rates entering each unit as follows:

$$\lambda_{U1} = P_{B1}\lambda$$
$$\lambda_{U2} = P_{B2}\lambda_{U1}$$
$$.....$$
$$\lambda_{Un} = P_{Bn}\lambda_{U_{n-1}},$$

Where, λ is the data rate entering the network device. $\lambda_{U1}, \lambda_{U2}, ..., \lambda_{Un}$ are the effective loads processed by the units. $P_{B1}, P_{B2}, ..., P_{Bn}$ are the blocking probabilities corresponding to each unit. Each unit will be treated as an M/G/1/K system. It means the performance measures, the mean vacation period, the mean service period and the energy gain are calculated for all units as presented in section 5.3.3 and depends only on the rate rejected by the previous unit without being dependent on the traffic matrix or sacrificing a huge computation time to define which ports to keep active. Also fault-tolerance is not affected since we did not change the routing paths. In fact, differently from other efforts described in section 2.5.4.5 in chapter 2, packets can reach their destinations from any routing path because only the method of relaying to the queue was changed in our system.

In this way, the data rate arriving to the network device can be treated by multiple

units. If the rate is not high, some units will stay inactive and consequently the energy can be gained.

5.4 System evaluation

Based on the metrics estimated in the section 5.3.3, we can now compare our new energy efficient system to the default system used in data centers (always-on devices).

5.4.1 Simulation environment

Throughout this simulation, we set the values of the mean service time $E[\sigma] = 1.5\ Tu$ (Time unit), the wake up time $T_w = 1\ Tu$, the listening time $T_l = 1\ Tu$ and the maximum queue size $K = 20$. Also, we suppose that the service times are exponentially distributed with parameter $\frac{1}{E[\sigma]}$.

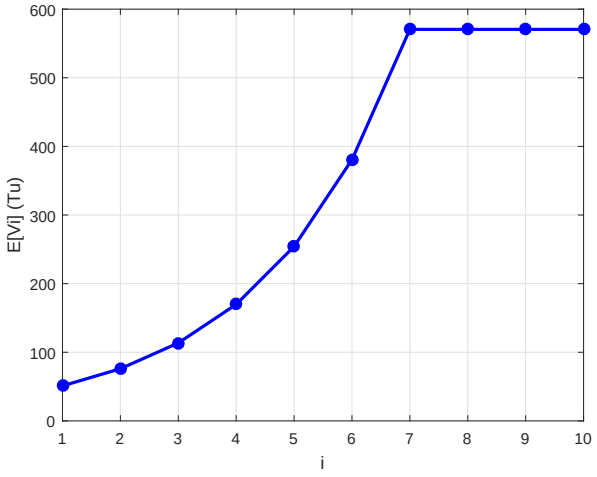


FIGURE 5.9: Enlargement of mean vacation times.

The first vacation time V_1 starts when all arrived packets are processed and the queue is empty. If during this vacation time, no arrival is detected, another vacation time initiates. We suppose that the probability density function of the vacation times $v(t)$ is exponentially distributed with parameter $\mu = \frac{1}{\bar{v}}$. Every vacation time $V_1, V_2, ..., V_N$ is also exponentially distributed with a mean size expressed as follows:

$$E[V_i] = \begin{cases} T_{min} + T_l & i = 1 \\ m^{min(i-1,l)} T_{min} + T_l & i = 2, 3, ... \end{cases}$$

Where, T_{min} is the initial vacation time size, m is the multiplicative factor to enlarge the next vacation times and l is the limit when the vacation time stops increasing. Figure

5.9 describes the enlargement of the vacation times, where $T_{min} = 50, m = 1.5$ and $l = 6$. The first mean vacation time V_1 is equal to T_{min}. The next mean vacation times sizes increase due to the multiplicative factor m. When $i > l$, the mean size of vacation times becomes uniform. This expression is used so that the mean vacation times keep enlarging if there are no arrivals until a certain threshold l to minimize the number of wake-ups.
The Laplace Stieltjes transform $L_{V_i}(\lambda)$ of V_i can be written as follows:

$$L_{V_i}(\lambda) = \begin{cases} \dfrac{T_l}{1 + T_{min}\lambda} & i = 1 \\ \dfrac{e^{-\lambda T_l}}{1 + m^{min(i-1,l)} T_{min}\lambda} & i = 2, 3, ... \end{cases}$$

5.4.2 Power saving strategy

In our power aware system, the amount of energy saved depends on 3 factors: the mean size of the vacation period where our processing units will be in sleeping state instead of wasting energy in idle state, the number of listening periods that should be optimized to detect rapidly the arrived packets and the size of queue which should be minimized in order to shorten the service period and return to vacation state. In our simulation, we will use the always-on system described in Figure 5.7 as the baseline and compare it to our proposed system depicted in Figure 5.6. We will take the ratio of energy gain (denoted by EG and expressed in Equation (5.41)) as the power conservation indicator. The values of the energy consumed in different states are summarized in Table 5.2.

TABLE 5.2: Energy consumed in different states of the processing unit [3].

Energy state	Value	explanation
C_{high}	5000^3 x 10^{-6} watt	Activity power
C_{low}	C_{high} x 0.7	idle power
C_{listen}	C_{high} x 0.3	listen and wake up power

5.4.3 Traffic indicator

To evaluate the performance of our system, we will assume that the traffic parameters (eg. data rate λ) are known, can be estimated or can be measured. The data rate λ is the average number of packets arriving per time unit *Tu*. A processing unit is working in high loads when $0.6 < \lambda \leq 1$. The data rate entering a network device is the sum of all

rates arriving to all ports. It means, in the always-on system, every port processes its received packets, however, in our system, the arriving data is merged in the controller level and treated depending on the availability of processing units as described in the previous sections.

5.4.4 Simulation results

In this section, we will solve a constrained optimization problem that maximizes the energy gain of the system while respecting a performance constraint. Then, we will use the optimized parameters to study, first, the performance of one processing unit in terms of packet waiting time, effective data rate and energy gain. Next, we will present the contribution of our system on power conservation in a network device and a data center network.

5.4.4.1 Constrained optimization problem

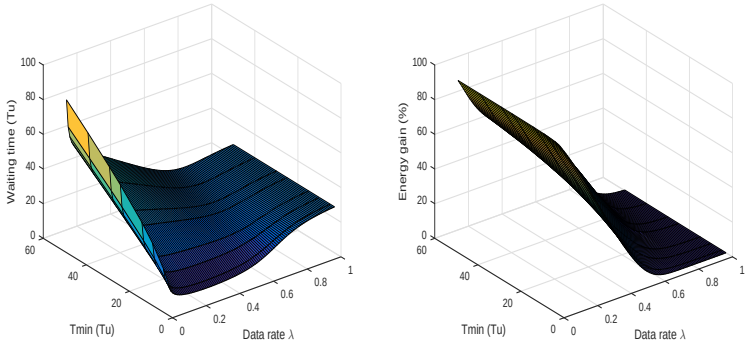
. Description of the problem

The evaluation of the expected energy gain and the mean waiting time are depicted in Figures 5.10, 5.11 and 5.12. To study the impact of T_{min} on the mean waiting time and the energy gained, we fix $m = 1.5$ and $l = 6$. Figure 5.10(a) presents the relation between T_{min} and $E[W]$ (mean waiting time) while Figure 5.10(b) depicts the relation between T_{min} and EG. As we can see, the size of the initial vacation time V_1 has an impact on the waiting time and the energy for different values of λ. More precisely, EG increases by the increase of T_{min}. For the mean waiting time $E[W]$, the contribution of T_{min} is visible only in low loads because the arrivals occur seldomly. So, the number of vacation times is big. However, the vacation period in high loads, generally, consists only of one vacation time which explains the hardly visible variation of the mean waiting time when increasing T_{min}.

To study the impact of the multiplication factor m, we fix $l = 6$ and $T_{min} = 30$. Similar to the impact of the initial vacation time size T_{min}, the factor to enlarge the next vacation times m contributes to increase the energy gain. However, it does not have a large effect on the waiting time in high loads due to the small number of vacation times, see Figures 5.11(a) and 5.11(b).

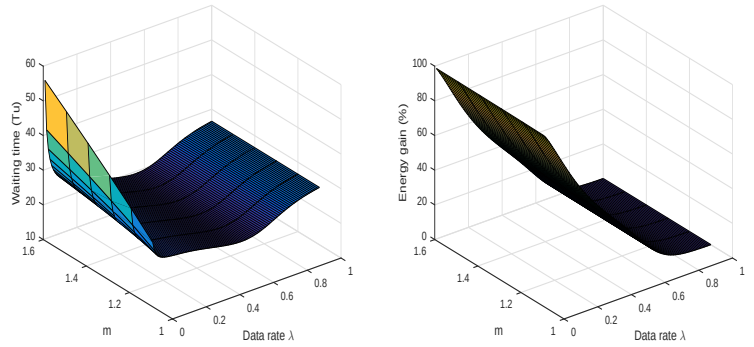
Finally, to study the impact of the limit factor l, we fix $m = 1.5$ and $T_{min} = 30$. The contribution of this factor is reflected in Figures 5.12(a) and 5.12(b). As we can see, l does not impact the energy gain for a fixed λ. However, it impacts the $E[W]$ in the low loads.

After these three observations, we can notice that when λ is small, the mean waiting



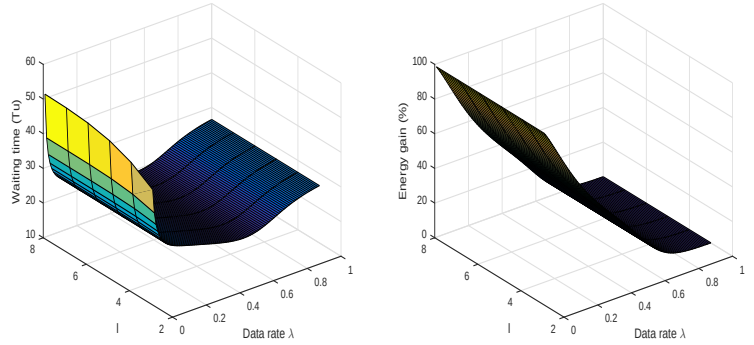
(a) (b)

FIGURE 5.10: Impact of T_{min} on the waiting time and energy gain.



(a) (b)

FIGURE 5.11: Impact of m on the waiting time and energy gain.



(a) (b)

FIGURE 5.12: Impact of l on the waiting time and energy gain.

time is large. This is explained by the fact that, in low loads, the occurrence of packets is not very frequent. Hence, since there is no arrivals, the number of vacation times is high and the last vacation size (V_N size) is big. So, the waiting time is bigger in low loads. In addition, when λ increases, $E[W]$ decreases rapidly, then, in high loads, it

increases again and becomes almost insensitive to the data rate. In fact, the mean waiting time is composed of two phases: the delay caused by the last vacation time V_N and the queuing delay when the processing unit is active. When the data rate λ increases, the last vacation time size decreases however the waiting in the queue increases since the number of arriving packets increases in high loads. In this way, the waiting time increases because of the increase of the queue size, then, the two phases balance each others leading to an insensitive waiting time in high loads. Concerning the energy gain EG, it decreases monotonically when the data rate increases. In fact, in high traffic loads, the idle time is short, hence, the vacation time is small and the gain is not very significant.

To conclude, the initial size of the vacation time T_{min}, the multiplication factor m and the limit of vacation enlargement l have a big impact on the waiting time and energy gain, especially in low loads. In addition, in low loads, our system achieves a high energy gain, however, this gain is accompanied with a large waiting time. In higher traffic loads, the energy gain is decreasing and the waiting time is proportionally short. Even though our system shows good results, it needs an optimization of the vacation parameters to minimize the mean waiting time in low packets arrival rates while maximizing the energy in higher rates even with an acceptable loss in terms of waiting time.

. Optimization

Beside our analytical model and theoretical results, we will try to solve an optimization problem in order to establish a trade-off between energy gain and system performance (mean waiting time). In this section, we will describe our multi-objectives formulation of the optimization. The performance objectives are the mean waiting time and the mean energy gain. We formulate this problem as a constrained optimization: the energy gain will be maximized while respecting an expected waiting time.

The parameters to optimize are T_{min}, m and l. Thus, we define our following program:

$$\begin{aligned} &\text{Maximize} \quad EG \\ &\text{subject to} \quad E[W] \leq W_{QoS} \end{aligned} \tag{5.43}$$

Where EG is given in equation (5.41), $E[W]$ is given in equation (5.32) and W_{QoS} is the waiting time constraint. The optimization in equation (5.43) maximizes the energy gain compared to the always-on system described previously or equivalently minimizes the energy consumed conditioned by a maximum of waiting time W_{QoS}. W_{QoS} is chosen by the administrator of the network. We have solved our constrained optimization for the parameters T_{min}, m and l with $W_{QoS} = 30$ and $W_{QoS} = 40$. The optimal parameters obtained when $W_{QoS} = 30$ are illustrated in Table 5.3 while the parameters obtained when $W_{QoS} = 40$ are shown in Table 5.4. We can see that the initial vacation time and

TABLE 5.3: Optimized parameters for $QoS = 30$.

λ	0.01	0.02	0.04	0.1	0.2	0.3	0.4	0.5	0.6
T_{min}	3	5	11	14	22	25	31	45	70.92
m	2	2	3	3	3	3.7513	4	4	4
l	4.82	8.06	1.56	9.96	6.99	9.94	9.98	6.57	10.12

TABLE 5.4: Optimized parameters for $QoS = 40$.

λ	0.01	0.02	0.04	0.1	0.2	0.3	0.4	0.5	0.6
T_{min}	5	7	13	15	25	30.7297	45.89	70.9	124.37
m	2.56	2.7	2.74	4.48	5	7	7	7	7
l	3	3	3	7.28	14.91	16.26	4.39	12.96	14.99

TABLE 5.5: Mean vacation time.

λ	0.01	0.02	0.04	0.1	0.2	0.3	0.4	0.5	0.6
$T_{min} = 50$, $m = 1.5$, $l = 6$	197.8	123.5	86.4	64.4	57.2	54.8	53.6	52.9	52.4
$T_{QoS} = 30$	150.6	100.5	57.9	41.3	35.8	36.5	40.3	52.6	77.5
$T_{QoS} = 40$	159.8	111.9	75.6	55.7	48.1	52.5	62.6	84.5	136

the multiplication factor are minimized in very low traffic loads in order to make the vacation period small and consequently make the waiting time shorter. The limit l is relatively big to keep the vacation times enlarging if there is no packets arriving. In higher traffic loads, the first vacation time T_{min} and the multiplication factor m are bigger compared to low loads in order to widen the waiting time while respecting the Qos (Quality of Service) constraint. In addition, for the same λ, the optimized parameters when $W_{QoS} = 40$ are bigger than the parameters when $W_{QoS} = 30$, which allows to have a bigger mean vacation period and consequently a bigger mean waiting time. Table 5.5 illustrates this fact and shows that for the case of $T_{min} = 50$, $m = 1.5$ and $l = 7$, the vacation period was big in the low load and decreases when the rate increases. However, after optimization, the vacation time is minimized for the low loads and it is maximized in bigger traffics to gain more energy with a little loss in terms of waiting time. We can see that with more concession in the system performance which means a larger waiting time constraint, a bigger vacation time is allowed. The performance in high loads ($0.6 < \lambda \leq 1$) is not optimized, since, even when enlarging the vacation period to maximize the energy gain, it is still negligible compared to the infinite service period (non-stop packets arrival).

5.4.4.2 Evaluation of one unit performance

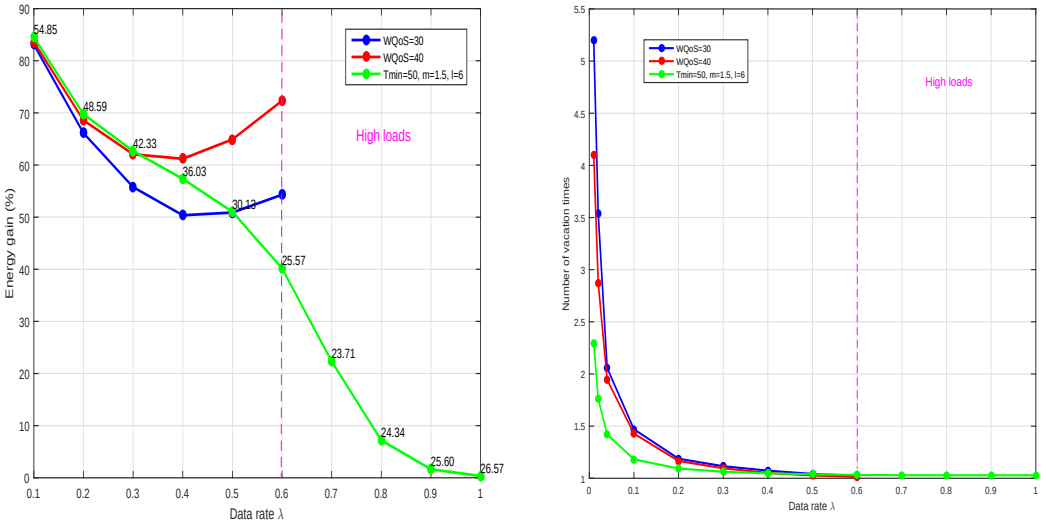
. Energy gain

Figure 5.13(a) exhibits the queuing system with vacations applied on one processing unit and shows the energy gain EG for different data rates. We presented EG for $T_{min} = 50$, $m = 1.5$ and $l = 7$ and we compared it to our optimization results when $W_{QoS} = 30$ and $W_{QoS} = 40$. The non-optimized mean waiting times are provided in the graph. We can see that the packets arrival rate has a great impact on the energy consumption. In fact, as shown in Figure 5.13(a), when increasing the network load, the power consumption increases and the power conservation decreases for the non-optimized parameters. This is explained by the fact that the non-optimized system only replaces the previously idle periods by vacation periods. It means our system accomplishes more power saving at the lowest network rates where the idle periods are large. We can see also, that the gain is almost proportional to the network load which allows our system to achieve the proportionality between the data rate and the energy consumption. However, as presented in the graph, the mean waiting time in low loads is large (superior than W_{QoS}) and can deteriorate the performance of the system. To solve this problem and enhance the performance of the system, the described optimization is conducted.

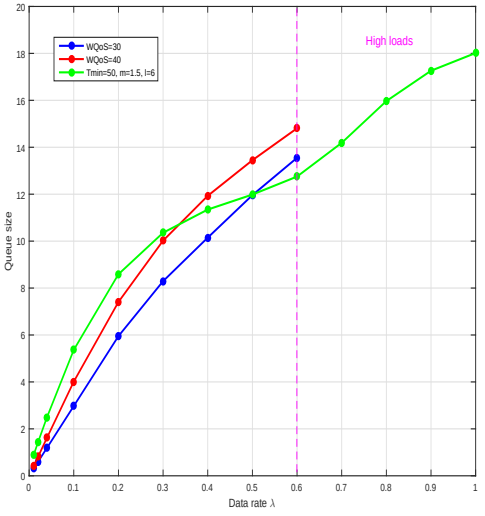
Three facts contributed to shorten the mean waiting time (respect the QoS constraint) in low traffic loads. The first one is the minimization of the vacation period as presented in Table 5.5. Another fact is the increase of the number of vacation times as presented in Figure 5.13(b) to listen periodically to the arrived packets and give an optimized sleeping period. Finally, our optimization contributed to reduce the queue size illustrated in Figure 5.13(c) and consequently reduce the waiting time in the queue in low loads. Enhancing the performance of the system in low loads is conditioned by sacrificing the gain (see Figure 5.13(a)). Yet, we still have a high energy gain reaching up to 70%. For the higher loads and since the network administrator can scarifies a little bit in terms of waiting time, the vacation period is widened. Thus, we obtained more than 50% of energy saved compared to the always-on system and we maximized the gain compared to the non-optimized system.

. Effective data rate

Figure 5.14 presents the effective data rate λ_U (see equation (5.15)) processed by the processing unit. The highest data rate is processed by the always-on system which is an expected result since every arriving packet is directly processed if the queue is empty. Without optimization, λ_U is smaller than the always-on system and the optimized systems, which is explained by the large vacation period. When we use the optimized vacation parameters, λ_U is approximately similar to the optimal system, in very low



(a) (b)



(c)

FIGURE 5.13: Contribution of the optimization in the energy gain.

loads. In higher traffic loads, the obtained effective rate is lower than the optimal one. However, this little performance loss is accepted in order to gain more energy.

5.4.4.3 Evaluation of multiple processing units performance

Figure 5.15 presents the effective data rate treated by each port in a 6-port device and by varying the rate arriving to the network device. As explained in the previous section, after the re-architecturing of the network devices, the first processing units will accept

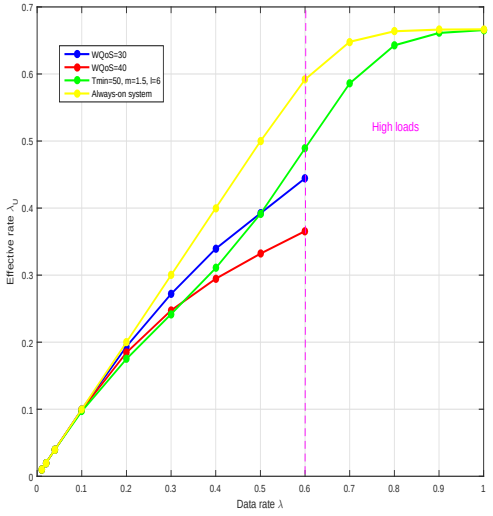


FIGURE 5.14: Effective data rate.

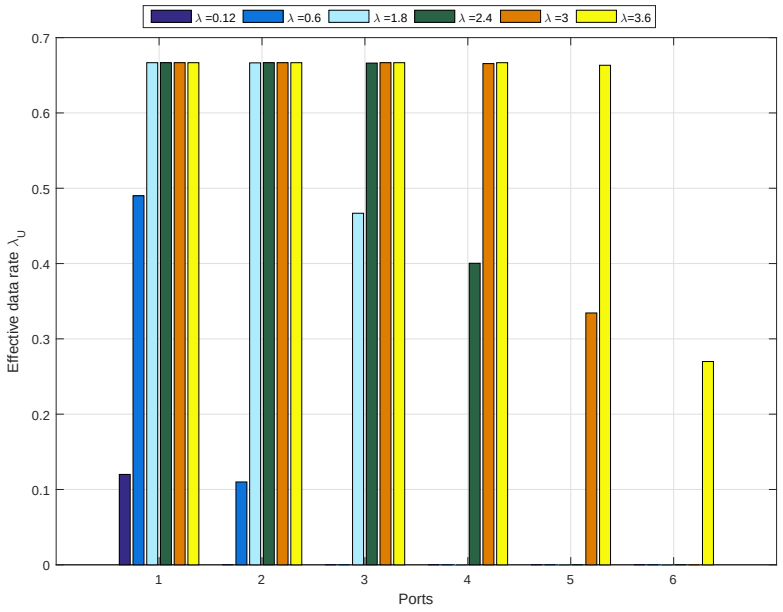


FIGURE 5.15: Distribution of data rates between ports when n=6.

higher load rates. However, the rest of the units will switch to sleep state or accept only low packets rates. In fact, the data rate λ is equal to the sum of all loads received by all the interfaces of the network device ($n = 6$). For example, if $\lambda = 1.8$, it means each interface received a load equal to 0.3. As we can see, for low loads, the received data is processed by only 1 to 3 units (see Figure 5.16). We can see also that, instead of activating all ports which is the case of the always-on system, our queuing model allows to treat the data by a minimum number of ports. In higher loads, when $\lambda = 3.6$, all units are active, however, the last one processes less load (see Figure 5.15). Hence,

our approach proved its efficiency to maximize the number of sleeping units without being dependent on the traffic matrix.

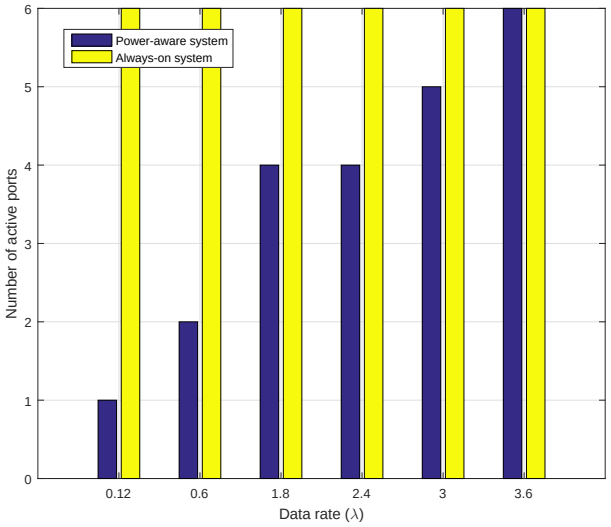


FIGURE 5.16: Number of active ports.

Figure 5.17 presents the energy gain of a 6-port device and a PTNet data center ($n = 6, s = 6$). By using our new proposed system and the packet scheduling algorithm based on queuing model, the energy gain can reach over 20%.

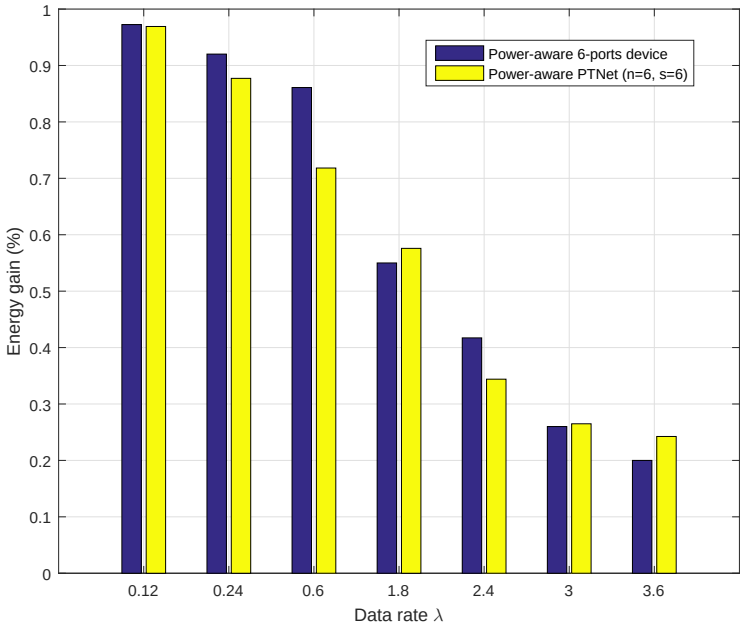


FIGURE 5.17: Energy Gain.

Figure 5.18 presents the packets drop rate of the power aware system compared to the always-on system. We can see that our approach decreases the drop rate which is explained by the fact that when the queue of one unit is congested, the packets are relayed to the next unit. However, in the case of the always-on system, a congestion leads to the packets loss.

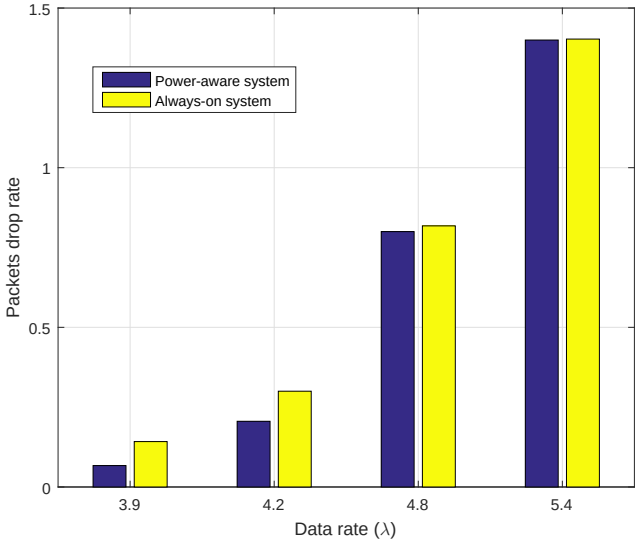


FIGURE 5.18: Drop rate.

5.5 Further discussion

In this section, we discuss some issues encountered when implementing our power aware approach, including choosing the constraints for the optimization and describing the drawbacks that should be handled in future works.

- Before implementing our proposed queuing system in a real data center, the administrator needs to choose the constraints for the optimization. Hence, he has to gather information about the data center, the latency needed to deliver packets and the requirements of the operators.
- Moreover, we can also optimize the queue length because it has a large impact on the congestion and blocking probability while respecting the maximum waiting time.
- Finally, our approach is deeply theoretically studied and proved a high power saving without depending on the traffic matrix. The implementation in a real data center using OpenFlow[84] is the subject of future works.

5.6 Conclusion

In this chapter, we studied the idea of decoupling the network device interfaces from their processing units. In this way, the incoming loads from different interfaces can be handled by only few available units. The other ones will be switched into sleep state. To maximize the number of sleeping units and manage the distribution of incoming packets, a Vacation/Service algorithm is proposed. Then, an analyze following the $M/G/1/K$ queuing model is conducted to estimate the energy gain, the vacation period, the service period, the waiting time, the effective data rate and the number of inactive units. Finally, an optimization is formulated to maximize the energy gain with a little concession in terms of waiting time. This approach grants a maximum power saving reaching more than 60 %. In addition, the competitive edge of this power-aware system is its non dependency on the traffic matrix and non complexity to compute the ports to keep active.

Chapter 6

Conclusion and future works

6.1 Conclusion

In the last few years, data centers become the backbone of the world business, economy, communication, and consumer services. Data centers offer a large scale and critical shared resources for huge number of users worldwide which creates a focal point for industrial and research efforts. In this thesis, we described some of the existing physical data center networks including the wired, wireless and optical infrastructures, we outlined existing power aware techniques and their feasibility for use in the DC environment, then, we presented a set of topology features that helped to estimate the data center performance and to establish a deep comparison between DC architectures.
By providing a deep analysis of data center research efforts, we classified some key challenges and drawbacks that need to be addressed including the high scalability, good fault tolerance, low latency, high capacity, load balancing, low cost, power efficiency and virtualization. Among these challenges, we chose, in this thesis, to address the problems of latency, scalability and power efficiency while guaranteeing the reliability of the system and a low cost of construction.
- Enhance the latency: Wireless technology (wFlatnet):
In order to present a solution to reduce the latency of packet transmission in data center networks, we proposed wFlatnet which is a novel design that integrates wireless shortcuts in Flatnet data center. This design creates shorter routes between servers by connecting devices wirelessly. wFlatnet has proved high performance in terms of fault-tolerance, diameter and average path length outperforming therefore other data center topologies.
- Enhance the scalability: Paramatrizable topology (PTNet):
The second addressed challenge is the scalability. In this context, we proposed PTNet

which is a new architecture highlighted by its parameterisable and gradual scalability. When implementing PTNet, the network administrator can choose the range size of its network without being obliged to implement a larger infrastructure. In addition to its gradual scalability, PTNet enjoys a low packet delivery delay and a robust fault tolerant algorithm. The implementation of the new network and the experimental results demonstrate its performance compared to well-known topologies.

- Greening the network: Power aware algorithms:

Finally, we addressed the problem of non proportionality between the traffic load and power consumption. In fact, we handled this challenge from two perspectives. First, from a routing perspective, we proposed two schemes where we keep active only the minimum number of network devices identified based on the traffic matrix. In addition, we made a reasonable trade-off between reliability, performance and power consumption. This approach grants a maximum power saving reaching up to 50%. However, it is dependent on the traffic matrix which is a random factor. Hence, we tackled the problem from a queuing perspective where we attributed a vacation period to the processing units of the network devices. Therefore, when the queue is empty, the unit can be set to sleeping mode. The energy wasted in idle mode can now be saved without being dependent on the traffic matrix.

6.2 Future works

The data center field is still in its infancy and it is rapidly evolving. However, despite the huge innovation in DCN research area handled by academia, big industrial companies and even small start ups in the last decade, this community has barely scratched the surface and there are still a large amount of challenges to be addressed. In this section, we will mention some existing related researches and a few open problems.

- Study of the interference in wireless DCs:

When integrating wireless links in data center networks, an interference investigation is needed and a channel allocation mechanism should be established. Eventually, we can strengthen our wFlatnet proposal by studying the possible interference between wireless links.

- Prediction of traffic matrix:

Another possible expansion to this work is to predict the traffic load arriving to the data center. In fact, since the unpredictability of the incoming traffic poses a barrier for the power saving, solutions to anticipate the future workloads and take the necessary power procedures are needed. In this context, artificial Intelligence techniques such as deep learning can be a potential method to predict the traffic loads based on past loads.

- Software-Defined Networking and security issues:

Security concerns are always posed as a major obstacle especially after the emerging of data centers in a highly virtualized environment. Currently, security solutions in data centers are hard to deploy, scale and manage. However, the Software-Defined Networking (SDN) which is a networking paradigm that decouples the control plan of a network from its forwarding plan has come with new attributes that are well suited for DC environment. In fact, the SDN centralized controller creates a single point of attack that is easy to control. Hence, recent research efforts are studying the integration of SDN in data centers and its contribution to enhance the network security. This research track can be a good topic to investigate in the future.

Appendix A

Appendix A

- Consider j=1,

$$\begin{aligned} F_1 &= \sum_{i=1}^{\infty} \int_0^{\infty} \frac{(\lambda t)^i}{i!} e^{-\lambda t} v(t) dt \\ &= \int_0^{\infty} (e^{\lambda t} - 1) e^{-\lambda t} v(t) dt \\ &= 1 - \int_0^{\infty} e^{-\lambda t} v(t) dt \end{aligned} \tag{A.1}$$

Using integration by parts:

$$\begin{aligned} \int_0^{\infty} e^{-\lambda t} v(t) dt &= (e^{-\lambda t} V(t)) \mid_0^{\infty} + \lambda \int_0^{\infty} e^{-\lambda t} V(t) dt \\ &= \lambda \int_0^{\infty} e^{-\lambda t} V(t) dt \end{aligned} \tag{A.2}$$

$$\int_0^{\infty} e^{-\lambda t} dt = -\frac{1}{\lambda} (e^{-\lambda t}) \mid_0^{\infty} = \frac{1}{\lambda} \tag{A.3}$$

Using equations (A.2) and (A.3), F_1 can be written as follows:

$$\begin{aligned} F_1 &= \lambda \times \frac{1}{\lambda} - \lambda \int_0^{\infty} e^{-\lambda t} V(t) dt \\ &= \lambda \int_0^{\infty} e^{-\lambda t} dt - \lambda \int_0^{\infty} e^{-\lambda t} V(t) dt \\ &= \lambda \int_0^{\infty} e^{-\lambda t} (1 - V(t)) dt \end{aligned} \tag{A.4}$$

The expression is verified for F_1.

Suppose that:

$$F_j = \lambda \int_0^{\infty} \frac{(\lambda t)^{(j-1)}}{(j-1)!} e^{-\lambda t} [1 - V(t)] dt \tag{A.5}$$

We have to prove that:

$$F_{j+1} = \lambda \int_0^{\infty} \frac{(\lambda t)^j}{j!} e^{-\lambda t} [1 - V(t)] dt \tag{A.6}$$

We can prove that:

$$\begin{aligned} &\int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t}[\lambda(1-V(t))+v(t)]dt \\ &= -(\frac{(\lambda t)^j}{j!} e^{-\lambda t}[1-V(t)])\mid_0^\infty + \lambda \int_0^\infty \frac{(\lambda t)^{(j-1)}}{(j-1)!} e^{-\lambda t}[1-V(t)]dt \\ &= \lambda \int_0^\infty \frac{(\lambda t)^{(j-1)}}{(j-1)!} e^{-\lambda t}[1-V(t)]dt = F_j \end{aligned} \tag{A.7}$$

$$\begin{aligned} F_{j+1} &= \sum_{i=j+1}^{\infty} f_i = \sum_{i=j}^{\infty} f_i - f_j = F_j - f_j \\ &= F_j - \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t} v(x)dt \\ &= \lambda \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t}[1-V(t)]dt \end{aligned} \tag{A.8}$$

- $$\begin{aligned} \sum_{i=1}^{\infty} F_j &= \sum_{i=1}^{\infty} \lambda \int_0^\infty \frac{(\lambda t)^{(j-1)}}{(j-1)!} e^{-\lambda t}[1-V(t)]dt \\ &= \lambda \int_0^\infty [1-V(t)]dt \\ &= \lambda[1-V(t)]t \mid_0^\infty + \lambda \int_0^\infty tv(t)dt \\ &= \lambda E[V] \end{aligned} \tag{A.9}$$

List of Publications

Journal papers

1. E. Baccour, S. Foufou, R. Hamila, Z. Tari and Albert Y. Zomaya, "PTNet: An Efficient and Green Data Center Network." Journal of Parallel and Distributed Computing (Elsevier).

2. E. Baccour, S. Foufou, R. Hamila and Z. Tari, "Achieving Energy Efficiency in Data Centers with a Performance-Guaranteed Power Aware Routing" Journal of computer communication (Elsevier).

3. E. Baccour, S. Foufou and R. Hamila, "A vacation queuing model analysis for greening data center networks" To be submitted.

Conference papers

4. E. Baccour, S. Foufou, R. Hamila and M.Hamdi, "A survey of wireless data center networks." CISS-15 International Conference on Information Science and Systems, March 2015.

5. E. Baccour, S. Foufou, R. Hamila and M.Hamdi, "wFlatnet: Introducing Wireless in Flatnet Data Center Network." Globecom-2015, December 2015.

6. E. Baccour, S. Foufou and R. Hamila, "PTNet: A parameterisable Data Center Network." 2016 IEEE Wireless Communications and Networking Conference, Doha, 2016, pp. 1-6.

7. E. Baccour, S. Foufou and R. Hamila, "A Guaranteed performance of a green data center based on the contribution of vital nodes." Globecom-2016, December 2016, Washington, DC USA.

8. E. Baccour, S. Foufou and R. Hamila, "An energy saving mechanism based on vacation queuing theory in data center networks" To be submitted.

Bibliography

[1] J.-Y. Shin, E. G. Sirer, H. Weatherspoon, and D. Kirovski, "On the feasibility of completely wireless datacenters," in *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2012, pp. 3–14.

[2] H. Vardhan, S.-R. Ryu, B. Banerjee, and R. Prakash, "60 ghz wireless links in data center networks," *Computer Networks*, vol. 58, pp. 192 – 205, 2014.

[3] C. Cheng, J. Li, and Y. Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," *Tsinghua Science and Technology*, vol. 20, pp. 28–39, 2015.

[4] Google data centers. [Online]. Available: https://en.wikipedia.org/wiki/Google_Data_Centers

[5] D. Jeffrey and G. Sanjay, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, pp. 107–113, 2008.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*. ACM, 2003, pp. 29–43.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*. USENIX Association, 2006, pp. 15–15.

[8] Y. Liu, J. Muppla, M. Veeraghavan, D. Lin, and M. Hamdi, *Data center network*. Springer, 2013.

[9] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. ACM, 2008, pp. 75–86.

[10] M. Jason Snyder, "Datacenter growth defies moore's law," in *http://www.pcworld.com/article/130921/article.html*, 2007.

[11] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 63–74, 2008.

[12] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*. ACM, 2009, pp. 63–74.

[13] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, 2010, pp. 17–17.

[14] K. Ramachandran, R. Kokku, R. Mahindra, and S. Rangarajan, "60 GHz data-center networking: Wireless => worry less?" *Technical report, NEC*, 2008.

[15] T. Wang, Y. Xia, J. Muppala, and M. Hamdi, "Achieving energy efficiency in data centers using an artificial intelligence abstraction model," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2015.

[16] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 68–73, dec 2008.

[17] (2008) Where are all the google data centers? [Online]. Available: https://techcrunch.com/2008/04/11/where-are-all-the-google-data-centers/

[18] Google data centers. [Online]. Available: https://www.google.com/about/datacenters/

[19] Microsoft global data centers. [Online]. Available: https://www.microsoft.com/en-us/cloud-platform/global-datacenters

[20] A. Froehlich. (2015) What are data center-class switches? [Online]. Available: http://searchnetworking.techtarget.com/feature/What-are-data-center-class-switches

[21] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: A survey," *Journal of Parallel and Distributed Computing*, pp. 45 – 74, 2016.

[22] Software-defined storage. [Online]. Available: https://en.wikipedia.org/wiki/Software-defined_storage

[23] "Best practices guide: Cabling the data center," *Technical report*, 2007. [Online]. Available: https://www.brocade.com/content/dam/common/documents/content-types/product-design-guide/cabling-best-practices-ga-bp-036-02.pdf

[24] Data center. [Online]. Available: https://en.wikipedia.org/wiki/Data_center

[25] L. B. Urs Hölzle, *The datacenter as a computer: An introduction to the design of warehouse-scale machines*. Synthesis Lectures on Computer Architecture, 2009, vol. 4, no. 1.

[26] Y. Liu and J. K. Muppala, "A survey of data center network architectures," 2013.

[27] T. Wang, Z. Su, Y. Xia, and M. Hamdi, "Rethinking the data center networking: Architecture, network protocols, and resource sharing," *IEEE Access*, vol. 2, pp. 1481–1496, 2014.

[28] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Computer Communications*, vol. 40, pp. 1 – 21, 2014.

[29] K. Bilal, S. U. R. Malik, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali, U. S. Khan, A. Abbas, N. Jalil, and S. U. Khan, "A taxonomy and survey on green data center networks," *Future Generation Computer Systems*, vol. 36, pp. 189 – 208, 2014.

[30] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," *SIGCOMM Comput. Commun. Rev.*, no. 4, pp. 51–62, 2009.

[31] Y. Sun, J. Cheng, Q. Liu, and W. Fang, "Diamond: An improved fat-tree architecture for large-scale data centers," *JCM*, no. 1, pp. 91–98, 2014.

[32] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "Ficonn: Using backup port for server interconnection in data centers," in *IEEE INFOCOM*, 2009, pp. 2276–2285.

[33] D. Lin, Y. Liu, M. Hamdi, and J. K. Muppala, "Flatnet: Towards a flatter data center network," in *GLOBECOM'12*, 2012, pp. 2499–2504.

[34] T. Wang, Z. Su, Y. Xia, Y. Liu, J. Muppala, and M. Hamdi, "Sprintnet: A high performance server-centric network architecture for data centers," in *Communications (ICC), 2014 IEEE International Conference on*, 2014, pp. 4005–4010.

[35] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly, "Symbiotic routing in future data centers," *SIGCOMM Comput. Commun. Rev.*, Aug. 2010.

[36] H. Vardhan, N. Thomas, S.-R. Ryu, B. Banerjee, and R. Prakash, "Wireless data center with millimeter wave network," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec 2010, pp. 1–6.

[37] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks." ACM SIGCOMM, 2009.

[38] W. Zhang, X. Zhou, L. Yang, Z. Zhang, B. Y. Zhao, and H. Zheng, "3d beamforming for wireless data centers," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM, 2011, pp. 4:1–4:6.

[39] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror mirror on the ceiling: Flexible wireless links for data centers," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. ACM, 2012, pp. 443–454.

[40] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*. ACM, 2010, pp. 327–338.

[41] Z. Chkirbene, M. O. Hasna, R. Hamila, and N. Hamdi, "Energy-efficient based on cluster selection and trust management in cooperative spectrum sensing," in *2016 IEEE Wireless Communications and Networking Conference*, 2016, pp. 1–6.

[42] C. Chauvenet, B. Tourancheau, and D. Genon Catalot, "Energy Evaluations for Wireless IPv6 Sensor Nodes," in *SENSORCOMM 2013, The Seventh International Conference on Sensor Technologies and Applications*, 2013, pp. 97–103.

[43] L. B. Saad, C. Chauvenet, and B. Tourancheau, "{IPv6} (internet protocol version 6) heterogeneous networking infrastructure for energy efficient building," *Energy*, vol. 44, no. 1, pp. 447 – 457, 2012.

[44] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing power consumption in backbone networks," in *Communications, 2009. ICC '09. IEEE International Conference on*, 2009, pp. 14–18.

[45] D. Çavdar and F. Alagoz, "A survey of research on greening data centers," in *Global Communications Conference (IEEE GLOBECOM)*, 2012, pp. 3237–3242.

[46] K.-K. Nguyen, M. Cheriet, M. Lemay, M. Savoie, and B. Ho, "Powering a data center network via renewable energy: A green testbed," *Internet Computing, IEEE*, vol. 17, pp. 40–49, 2013.

[47] M. Arlitt, C. Bash, S. Blagodurov, Y. Chen, T. Christian, D. Gmach, C. Hyser, N. Kumari, Z. Liu, M. Marwah, A. McReynolds, C. Patel, A. Shah, Z. Wang, and R. Zhou, "Towards the design and operation of net-zero energy data centers," in *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 13th IEEE Intersociety Conference on*, 2012, pp. 552–561.

[48] Google green. [Online]. Available: http://www.google.com/green/bigpicture/references.html

[49] (2011) Open compute project in practice. [Online]. Available: http://opencompute.org/about/energy-efficiency

[50] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*. ACM, 2011, pp. 31–40.

[51] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. ACM, 2009, pp. 37–48.

[52] S. U. Khan and A. Y. Zomaya, Eds., *Handbook on Data Centers*. Springer, 2015.

[53] X. Guan, B.-Y. Choi, and S. Song, "Energy efficient virtual network embedding for green data centers using data center topology and future migration," *Computer Communications*, vol. 69, pp. 50 – 59, 2015.

[54] Vmware infrastructure architecture overview. [Online]. Available: http://www.vmware.com/

[55] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "Greencloud: A new architecture for green data center," in *Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session*. ACM, 2009, pp. 29–38.

[56] D. Arai and K. Yoshihara, "Eco-friendly distributed routing protocol for reducing network energy consumption," in *Eco-friendly distributed routing protocol for reducing network energy consumption*, 2010, pp. 25–29.

[57] D. Ferguson, A. Lindem, and J. Moy, "Ospf for ipv6," jul 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5340.txt

[58] K. H. Ho and C. C. Cheung, "Green distributed routing protocol for sleep coordination in wired core networks," in *Networked Computing (INC), 2010 6th International Conference on*, 2010, pp. 11–13.

[59] A. Carrega, S. Singh, R. Bolla, and R. Bruschi, "Applying traffic merging to datacenter networks," in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, 2012, pp. 1–9.

[60] Y. Shang, D. Li, and M. Xu, "Energy-aware routing in data center network," in *Proceedings of the First ACM SIGCOMM Workshop on Green Networking*, 2010, pp. 1–8.

[61] I. Adan and J. Resing, *Queueing Systems*, 2015. [Online]. Available: http://www.win.tue.nl/~iadan/queueing.pdf

[62] S. Fang, H. Li, C. H. Foh, Y. Wen, and K. M. M. Aung, "Energy optimizations for data center network: Formulation and its solution," in *Global Communications Conference (IEEE GLOBECOM)*, 2012, pp. 3256–3261.

[63] C. Schwartz, R. Pries, and P. Tran-Gia, "A queuing analysis of an energy-saving mechanism in data centers," in *The International Conference on Information Network 2012*, 2012, pp. 70–75.

[64] E. Baccour, S. Foufou, R. Hamila, and M. Hamdi, "A survey of wireless data center networks," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, 2015, pp. 1–6.

[65] S. Pinel, P. Sen, S. Sarkar, B. Perumana, D. Dawn, D. Yeh, F. Barale, M. Leung, E. Juntunen, P. Vadivelu, K. Chuang, P. Melet, G. Iyer, and J. Laskar, "60 GHz single-chip cmos digital radios and phased array solutions for gaming and connectivity," *IEEE J.Sel. A. Commun.*, pp. 1347–1357, 2009.

[66] E. Baccour, S. Foufou, R. Hamila, and M. Hamdi, "wflatnet: Introducing wireless in flatnet data center network," in *2015 IEEE Globecom Workshops (GC Wkshps)*, 2015, pp. 1–6.

[67] A. M. Niknejad and H. Hashemi, *mm-Wave Silicon Technology 60 GHz and Beyonds*, 2008.

[68] E. Baccour, S. Foufou, R. Hamila, and M. Hamdi, "A survey of wireless data center network." *Proceeding of CISS-15 International Conference on Information Sciences and Systems*, March 2015.

[69] Ns-3. [Online]. Available: https://www.nsnam.org/documentation/

[70] B. Theophilus, A. Ashok, A. Aditya, and Z. Ming, "Understanding data center traffic characteristics," *SIGCOMM Comput. Commun. Rev.*, pp. 92–99, 2010.

[71] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A cost comparison of datacenter network architectures," in *Proceedings of the 6th International Conference*. ACM, 2010, pp. 16:1–16:12.

[72] E. Baccour, S. Foufou, and R. Hamila, "Ptnet: A parameterizable data center network," in *2016 IEEE Wireless Communications and Networking Conference*, 2016, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7564868/

[73] "Flatten your data center architcture-white paper." Juniper networks, 2011. [Online]. Available: http://www.juniper.net/search/public/

[74] S. Morgan, "Cisco aci fabric simplifies and flattens data center network," in *http://blogs.cisco.com/ciscoit/b-aci-08012014-cisco-aci-fabric-simplifies-data-center-network*, 2014.

[75] T. Wang, Y. Xia, D. Lin, and M. Hamdi, "Improving the efficiency of server-centric data center network architectures," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 3088–3093.

[76] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 350–361, 2011.

[77] T. Wang, Y. Xia, J. Muppala, M. Hamdi, and S. Foufou, "A general framework for performance guaranteed green data center networking," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, 2014, pp. 2510–2515.

[78] P. Delforge, "America's data centers consuming and wasting growing amounts of energy," National resources defense council, Tech. Rep., 2015.

[79] J. Koomey, "Growth in data center electricity use 2005 to 2010," Oakland, CA: Analytics Press, Tech. Rep., 2011.

[80] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12. ACM, 2012.

[81] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*. ACM, 2010, pp. 338–347.

[82] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, *A Power Benchmarking Framework for Network Devices*. Springer Berlin Heidelberg, 2009.

[83] S. Kanev, K. Hazelwood, G. Y. Wei, and D. Brooks, "Tradeoffs between power management and tail latency in warehouse-scale applications," in *2014 IEEE International Symposium on Workload Characterization (IISWC)*, 2014, pp. 31–40.

[84] Openflow. [Online]. Available: http://archive.openflow.org/

[85] Cisco ios netflow. [Online]. Available: http://www.cisco.com/web/go/netflow

[86] E. Baccour, S. Foufou, R. Hamila, Z. Tari, and A. Zomaya, "Ptnet: An efficient and green data center network," *Journal of Parallel and Distributed Computing*, 2017.

[87] E. Baccour, S. Foufou, and R. Hamila, "A guaranteed performance of a green data center based on the contribution of vital nodes," in *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1–6.

[88] E. Baccour, S. Foufou, R. Hamila, and Z. Tari, "Achieving energy efficiency in data centers with a performance-guaranteed power aware routing," *Computer Communications*, 2017.

[89] L. Smith, Ed., *A Tutorial on Principal Components Analysis*, 2002.

[90] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill, 2001, ch. 22.

[91] D. S. V. Medeiros, M. E. M. Campista, N. Mitton, M. D. de Amorim, and G. Pujolle, "Weighted betweenness for multipath networks," in *2016 Global Information Infrastructure and Networking Symposium (GIIS)*, Oct 2016, pp. 1–6.

[92] H. Kim and K. P. Kim, "Formulating closeness centralities on workflow-supported performer-activity affiliation networks," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, Feb 2017, pp. 960–965.

[93] A. P. Bianzino, "Energy aware traffic engineering in wired communication networks," Theses, Télécom ParisTech ; Politecnico de Turin, May 2012. [Online]. Available: https://pastel.archives-ouvertes.fr/pastel-01002105

[94] R. Froom and E. Frahim, *Implementing Cisco IP Switched Networks (SWITCH) Foundation Learning Guide: (CCNP SWITCH 300-115)*. Pearson Education, 2015.

[95] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," 2015, pp. 27–51.

[96] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "Netfpga–an open platform for gigabit-rate network switching and routing," in *Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education*, 2007, pp. 160–161.

[97] J. Kingman, *Poisson Processes*. Clarendon Press, 1992.

[98] S. K. Bose, *An Introduction to Queueing Systems*. Springer, 2002.

[99] H. Takagi, "M/g/1/k queues with n-policy and setup times," *Queueing Systems*, vol. 14, pp. 79–98, 1993.

[100] S. Alouf, E. Altman, and A. P. Azad, "Analysis of an M/G/1 queue with repeated inhomogeneous vacations with application to ieee 802.16e power saving mechanism," in *2008 Fifth International Conference on Quantitative Evaluation of Systems*, 2008, pp. 27–36.

[101] E. Chromy, T. Misuth, and M. Kavacky, "Erlang c formula and its use in the call centers," in *Information and communication technologies and services*, 2011.