optimizing routing and resource allocation in sdn-based server only data center networks for private cloud architectures

Roua Touihri

UNIVERSITÉ PARIS-EST

devoteam

Paris-Est University
MSTIC Doctoral School

Presented by

Ms. Roua TOUIHRI

Submitted for the degree of

Doctor of Philosophy from University of Paris-Est

Title:

Optimizing Routing and Resource Allocation in SDN-based Data Center Networks for Private Cloud Architectures

Committee:

Mrs. Anne FLADENMULLER	*Reviewer*	*Associate Professor – HDR, University of Sorbonne*
Mr. Pascal LORENZ	*Reviewer*	*Full Professor, University of Haute Alsace*
Mr. Yassin AMIRAT	*Examiner*	*Full Professor in University of Paris Est*
Mr. Ken CHEN	*Examiner*	*Full Professor, University of Paris 13*
Mr. Marouane MECHTERI	*Examiner*	*Virtualization Architect at Bouygues Telecom*
Mr. Abdulhalim DANDOUSH	*Advisor*	*Associate Professor, ESME-Sudria, Paris Sud*
Mr. Cyril VEILLON	*Advisor*	*Devoteam Research and Innovation Manager*
Mr. Nadjib AITSAADI	*Supervisor*	*Full Professor, UPEC/UVSQ Paris-Saclay*

To my lovely Hwita.
To my dear dad, brother and sister.

Acknowledgment

Foremost, I would like to thank my supervisor, Prof. Nadjib AITSAADI for his support and important recommendations.

I would like to express my deep gratitude to Devoteam and particularly to my manager M. Cyril VEILLON for his guidance during this extraordinary experience within the Devohub team and for his outstanding guidance and support.

Besides, I would like to thank my co-advisor Dr. Abdulhalim DANDOUSH for his continuous support and precious advice.

Special thanks go to all jury members for reviewing my thesis, examining my defense, in addition to their constructive and relevant comments, remarks, and questions.

Moreover, I would like to thank both my Devoteam and laboratory colleagues and mates for the friendly working atmosphere, our interesting and rich conversations, and particularly for their encouragement and positive energy. I would like to express my deep acknowledgment to Safwan ALWAN, Olivier RABILLOUD, Chemseddine MENAKBI, Thibault CHAILLOUX, Kavuthen KUNAPALAN, Boutheina DAB, Ons FARES, Meriem AHMED BACHA, and Salma REBAI.

I am sincerely grateful to our family's friend and CEO of Neoledge, M. Mongi ZIDI who gives me the opportunity to successfully begin my professional career by allowing me to join a motivating and innovative team. Undoubtedly, this experience played a key role in participating in my professional achievements of today.

Last but not least, I would like to express my absolute gratitude and deep love to my parents, Mekki and Hayet, my lovely brother and sister, Raed and Rania. Words cannot really describe my deep and sincere acknowledgment to them. But, I would like to say that they are and will represent my biggest strength, pride, and happiness.

Abstract

According to the latest statistics, the number of connected people to the Internet is still exponentially growing and the high quality of cloud services will remain significantly requested in the coming years. Facing the resulting tremendous growth of the intra-data center traffic, the traditional Data Center Network (DCN) architectures are not capable to stay ahead of the demand in terms of scalability and cost lowering. Besides, DCNs suffer from the degradation of their network Quality of Service (QoS) performances which deeply impact the users Quality of Experience (QoE). As a result, cloud architects and service providers are pressed to deal with this rapid explosion of data center traffic by reconsidering their data center architecture to fit their customers' needs. Hence, implementing new techniques based on optimized algorithms become compulsory to overcome these challenges within data center networks. In this regard, we tackle the performance of data center networks by investigating new promising approaches able to improve intra-data center communication and to optimize the resource allocation within their infrastructures. Motivated by its architecture strengths, we study in this thesis the CamCube Server-Only data center managed by ONOS SDN controller. Besides, we aim to implement and analyze the performance of strategies to address the optimization of routing and resource allocation for intra-CamCube servers communications. In fact, our objective consists of improving network performance and decreasing network congestion. The problem of resource allocation and routing in SDN-based CamCube topology is NP-hard. In order to overcome this challenge, we investigate the problem through three increasing complexity steps. In the first stage, we consider unicast intra-CamCube communication by proposing new methods named `CRP` and `ACO-CRP` respectively based on Linear Programming Ant Colony Optimization. The latters generate an optimized path to forward packet in a typical CamCube network subjects to the constraints of network performance in terms of latency and path length. In the second phase, we propound a new `M-CRP` and `ACO-MCRP` approaches to tackle the multicast communications in SDN-based CamCube network. Our approaches addressed multicast routing and resource allocation of online arrived flows. We show that our propounded approach enhanced the CamCube QoS and the quality of the proposed multicast tree. In this thesis, we online treat the routing and the resource allocation for the communication of each arrived flow. As a third step, we focused on resource allocation optimization for batch mode arrival of traffic flows within ONOS-based CamCube topology by proposing `batch-(M)CRP` strategy. Note that we emulated the propounded environment to test the full proposed schemes, through extensive experimentations conducted with Mininet. We evaluate the performance of our propositions in terms of E2E delay, jitter and packet loss. Obtained results demonstrate that our proposals outperform the existing state-of-the-art strategies such as the shortest path and `OSPF` routing schemes respectively for CamCube and traditional Clos DCNs architectures.

Key words:

DCN, CamCube, SDN, Routing, Resource allocation, Optimization.

Résumé

Selon les dernières études scientifiques, l'augmentation du nombre des utilisateurs connectés à internet suit encore une tendance exponentielle exigeant toujours une excellente qualité de services (QoS) Cloud. Les architectures traditionnelles des centres de données existantes sont incapables de suivre cette croissance importante en termes de scalabilité et d'optimisation des coûts, et la dégradation de la QoS de leurs réseaux qu'elles subissent, impactent la Qualité d'Expérience des utilisateurs. Par conséquent, les opérateurs du cloud se pressent de travailler sur l'amélioration de l'architecture de leurs centres de données afin de répondre aux exigences de leurs clients. De fait, mettre en place de nouvelles architectures physiques et/ou implémenter de nouvelles techniques plus performantes devient un impératif pour relever ces défis. Dans ce contexte, nous proposons, dans le cadre de cette thèse, une nouvelle approche prometteuse afin d'adresser le problème de l'amélioration du routage et de l'optimisation de l'allocation des ressources dans les réseaux au sein même des centres de données. Nous proposons une architecture centralisée qui se base sur le Software Defined Networking (SDN) pour gérer le réseau au sein du centre de données CamCube composé uniquement de serveurs afin de traiter les données et relayer les paquets aux autres voisins dans la même topologie. Le problème du routage et de l'allocation de ressource dans la topologie CamCube est NP-difficile. Afin de passer outre cette complexité, nous proposons dans ces travaux de thèse de procéder suivant trois étapes. Dans un premier temps, nous considérons le problème de génération de flux unicast entre une source et une destination. Nous proposons pour cela deux nouveau protocoles `CRP` qui se base sur la programmation linéaire et la meta-heuristic `ACO-CRP` inspirée de l'algorithme Ant Colony Optimization. Ces deux protocoles proposent un chemin optimal entre la source et la destination améliorant la performance du réseau CamCube en termes de temps de latence, de perte de paquets et de gigue. Dans un deuxième temps, nous adressons le problème de routage multicast dans le réseau SDN CamCube. Nous proposons deux nouvelles approches `M-CRP` `ACO-MCRP` dont le but de l'amélioration non seulement de la QoS du réseau CamCube mais aussi de l'arbre multicast. Ces différentes propositions pour les flux unicast et multicast traitent le routage et l'installation des flux arrivées en mode en ligne. En troisième étape, nous proposons le protocole `Batch-CRP` qui gère conjointement le routage et l'allocation de ressources pour un arrivage en bloc de flux i.e., suivant le mode batch. Finalement, après différentes émulations effectuées en utilisant Mininet et le contrôleur ONOS, nous montrons que les protocoles proposés sont plus performants que les stratégies existantes dans la littérature tels que les protocoles du plus court chemin et OSPF respectivement dans CamCube et Clos centres de données.

Mots-clès :

DCN, CamCube, SDN, Routage, Allocation de ressource, Optimisation.

Table of contents

Chapter 1
Introduction

Contents

1.1 Data Center Transformation in the Cloud Computing

Since the late of 1950s, IBM and American Airlines [1] expressed the need of providing more availability of passengers and reservations data accessing to any agent and at any location. Hence, the concept of data centers appeared. Lately, the idea became a reality and have been improved by adding new features to develop data centers infrastructure. Furthermore, the data center fabric has been moved from the yesterday's mainframe to the modern cloud scalable fabrics.

Later in 1990s, Data centers became famous as a service model [2]. More companies such as Apple and VMWare treated compatibility issues by improving the Virtual PC. Afterwards, VMWare launched the bare-metal hypervisor [3] where applications can be executed without an additional underlying operating system. Since 2000, with the exponential growth of cloud applications, several companies such as Amazon have introduced the Cloud Computing Services. Indeed, these companies move their IT infrastructure running services to businesses by designing the "web services"[2].

Since 2007, the cloud computing has promised new features and services such as dynamic IT infrastructure based on data centers, customable software services and insuring computing environment QoS [4]. This emerging paradigm proposes basically three main services [5]: i) *Infrastructure as a Service* (*IaaS*) where IT infrastructure is provided to customers such as storage, processing and networks. It allows to the consumer to run and deploy the provided software including operating system and applications. For this offered service, the client does not have the control on the cloud infrastructure. Indeed, the provided IT infrastructure is fully maintained and administrated by the cloud provider. ii) *Platform as a Service* (*PaaS*) where consumer can deploy needed application based on programming, libraries, tools and services through the cloud infrastructure. These application supported by the provider are managed and configured by the client. However, the latter cannot control the underlying cloud infrastructure including network, servers, storage, etc. iii) *Software as a Service* (*SaaS*) allows to cloud consumer to execute and use the provider's application. Clients can access these cloud applications offered as a provider's service through different devices. The cloud application can be available through web browser or program interface. Clients using this type of cloud services can have a limited control on the provided application settings. In fact, providers are responsible of the maintenance and the management of the infrastructure, the cloud platform hosting these services and of the latter's configuration and availability. Figure. 1.1 details the interaction between the basic cloud computing services and actors. Besides, the cloud computing offers different deployment model [5]: i) *Private Cloud* where the infrastructure is dedicated for an exclusive use of an organization composed by multiple consumers (e.g. company, university, etc.). The

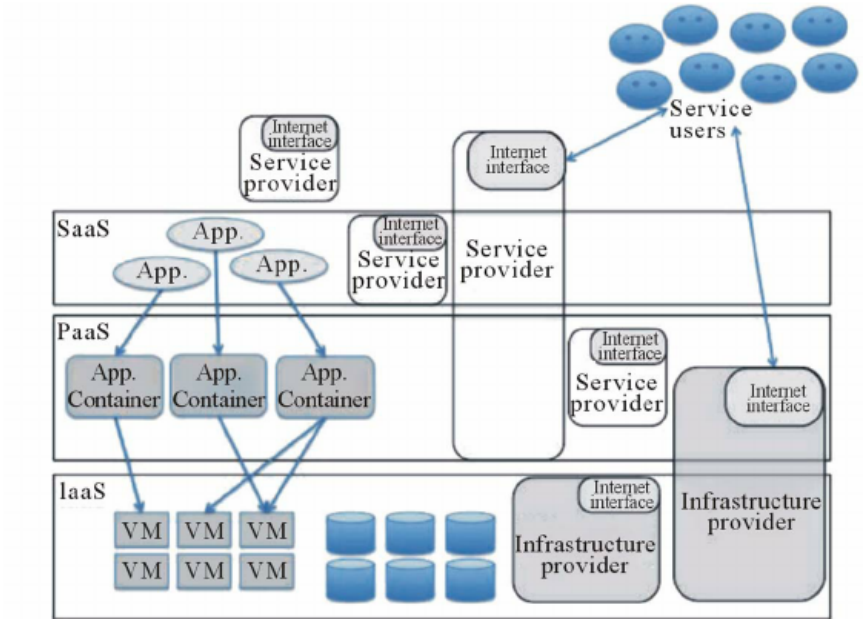

Figure 1.1 – Cloud Computing Services and Actors [6]

cloud infrastructure is managed and operated by the organization. ii) *Public Cloud* where the cloud infrastructure is accessible to anyone. It can be owned and managed by a company, a government or academic organizations or a combination of them. iii) *Community cloud* : it is a cloud shared by a particular community from organization with a common concern such as mission, security requirement, etc. Finally, iv) *Hybrid Cloud* which is a combination of two or more of public and private cloud infrastructures (private, community and public). Nowadays, the cloud's popularity resides on its characteristics such as on-demand self-service, fast elasticity, resource pooling by serving cloud user via several tenants and measured service where service usage can be controlled and monitored by cloud administrator.

Furthermore, in 2012, 38% of businesses were using the cloud and 28% of other companies had the plan to initiate or expand their use of the emerging concept [1]. Recently, this tendency rocketed with 90% of companies using cloud services. In 2017, 45% of workloads was running on hosted cloud services and this percentage rose to 60% in 2019 [7]. Also, the distribution of worldwide cloud services workload will move from 62% in 2018 to reach 83% in 2020 compared to the traditional resources and infrastructures [8].

Consequently, the increasing interest for cloud services, pushes the biggest companies to move their data from small physical severs in their buildings to huge data center buildings often distributed all over the world. This rapid IT infrastructure transformation gives birth to today's cloud titans: Google, Apple, Facebook, Amazon and Microsoft, also called GAFAM. The latters

cloud providers account for 61% of the total market in 2018 according to Canalys Report [9]. Amazon still the leader with 32% followed by Microsoft Azure participating with 17%. Also, experts expect a sustained growth of worldwide cloud infrastructure services investments from only 80 US$ billion in 2018 to surpass 143 US$ billion in 2020 [9].

Moreover, according to a recent Gartner's report [10], the worldwide public cloud services market revenue will grow from 227.8 US$ billion in 2019 to 266.4 US$ billion in 2020. Internet actors like Google, Microsoft and Amazon boost their investment in designing Data centers. In fact, the Capital Expenditure (CAPEX) has increased from 9.7% in 2016 to 55% in 2018 [11]. Besides, according to Cisco Global Cloud Index [12], traffic within hyper-scale data centers will attend 55% by 2021. Thus, traffic volume will quadruple. The latter behavior will pose new challenges for researchers and industrial in order to design new efficient and scalable data center architectures. But, recently, Data center fabrics start suffering from their own success. In fact, it becomes very hard to handle the increasing number of cloud services while providing a high performed content delivery to final customers. Hence, guaranteeing a high quality of service still an important challenge that cloud architects aim to alleviate it especially when their objective is to minimize latency while ensuring high computing performance within data center infrastructures.

Afterwards, the arrival of 5G cellular technology and the explosion of the number of mobile devices can impact the development of mobile network technologies. Consequently, experts propound to not only insure a high performed services with high computing power but also providing fast access speed. Hence, improving data centers architectures is no more enough to handle with these new challenges especially that existing cloud computing architecture suffer from several limitations [13]. First, cloud infrastructures encounter a user access problem impacted by the high propagation distance. Indeed, because of the far located physical servers, the transmission of exchanged data between the mobile device and the requested data center should take a long distance. This behavior leads to increase the transmission latency especially for data forwarded across various type of cloud architectures e.g., wireless networks. Thus, the cloud performance and the QoS of the accessed network can be impacted and suffer from a considerable degradation. Second, the cloud computing designs have to deal with resource management problem because of the compact dimension of the mobile and Internet of Things (IoT) devices to run highly consumed applications with a needed voluminous storage such as image analysis, real time games, etc. Finally, the existing cloud architectures relies on data center infrastructure to run and process data to users thanks to the installed and highly performed servers within it. Unfortunately, the increasing number of mobile users concentrated in a given network infrastructure can engender a long latency service with improperly allocated computing resources. Therefore, the Mobile Edge Computing (MEC) is one of the solutions that has

emerged.

Indeed, the MEC is acknowledged by the 5G Infrastructure Public Private Partnership (5G PPP) in addition to the Network Functions Virtualization (NFV) and Software Defined Networking (SDN) [14]. The MEC is considered as an evolution of mobile base stations in addition to the union of both IT platform and telecommunications networking. Hence, this emerging technology will represent a key architectural concept to improve the evolution of the 5G. Particularly, the MEC technology improves the evolution of the mobile broadband network by introducing the programmable features. It also outfits the 5G architecture requirements by ensuring a high throughput, scalability, automation and a minimized latency.

Actually, the MEC is standardized by the ETSI Industry Specification Group (ISG) [15]. The newly propounded technology is based on virtualized platform where it performs both IT environment and cloud capabilities in the Radio Access Network (RAN) and ensures the adjacent proximity of user subscribers within the mobile network [15]. Since the platform hosting MEC and NFV technologies are similar, it is recommended to reuse the NFV framework and its infrastructure management to run both MEC applications and Virtual Network Functions VNFs on a common platform[15]. Furthermore, the MEC design consists on locating close servers to users location in order to ensure both powerful computing services and a very high quality of service (by extremely minimizing the content delivery latency). In fact, the MEC technology aims to maximize the network services availability to a mobile operator customers and offers a high bandwidth, a low latency, a good proximity, real-time radio network information and location alertness. Consequently, thanks to the above advantages provided by the MEC, mobile operators consider it as an opportunity to improve their mobile broadband users experience. Also, content providers can easily deliver their critical applications over the mobile network[15]. Therefore, MEC is considered as a promising technology that ensures a new business opportunities for all chain actors (mobile operators, enterprises, etc.) which will impact the business market growth and create new use cases for several sectors [15].

Considering the recent arrival of MEC technology, few studies tackled it. We can cite [16] [17] [18], the authors make a survey of the emerged technology by introducing some recent studies and typical MEC applications. Also, as an example of MEC proposed architecture, we can cite [19] where the authors presented an hierarchical scheme that aims to strike a balance between the rapid content delivery and the high computed services based on context awareness. Moreover, with the emergence of IoT applications, machine learning and multimedia streaming, the MEC technology promises a substantial performance benefits to these newly born applications that require ultra-low service latency, highly computed programs and a reduced energy consumption. These new challenges motivate academia and industry to focus their efforts on proposing real implemented architecture in the incoming years.

In this regards, cloud architects focus their studies on designing a high performed data center architectures to meet actual business needs and to meet the challenges of cloud application intensive workload. However, only improving data center architecture is no more enough to handle with these new challenges and requirements. Therefore, it is also important to address the flow routing and communication for intra-data center traffic in order to improve the performance of its network and then to satisfy the required Quality of Service for cloud users.

In the next sections, we will introduce the main data center design. Then, we will expose the main challenges of these fabrics. Later, we will define the addressed problem of this PhD thesis. Afterwards, we will summarize the main contributions.

1.2 Data Center Network Design

In the recent years, the cloud computing has largely impacted our way of life. In fact, this paradigm provides a service model based on providing on-demand resources following a specific Service-Level Agreement (SLA) required by the cloud stakeholders. Also, the cloud computing owe its success to the highly performed infrastructure in the data center. Indeed, the latter presents a crucial element for a stable and efficient cloud services. However, to deal with the tremendous volume of traffic distributed over a very high number of servers, it becomes compulsory to focus on data center design in which cloud operators can efficiently manage this traffic while alleviating the high maintenance cost meanwhile.

Data centers are considered as a warehouse-scale computer [20] where a hundred even hundred of thousands of servers are hosted to build the cloud infrastructure. Some companies deploy several data centers geographically distributed and interconnected to each other to form it own cloud infrastructure. for example, in order to enhance IT infrastructure efficiency, Facebook continue scaling its data-center by expanding it to 15 data center locations in 2018 [21]. Also, in order to address the issue of scalability, companies work on designing and improving their IT infrastructure to deal with varied traffic patterns by installing powerful servers and re-organize the data center racks distribution.

Actually, Cloud architects address the ecological issues and the energy-efficiency data center consumption. In fact, some data center can exceed the national energy consumption of some countries by using an estimated of 200 terawatt hours each year [22]. So, companies share their commitment to build new data center building based on renewable energy. For instance, Facebook aims to use 100% renewable energy and reduce by 75% its greenhouse gas emission by 2020 [21].

1.2.1 Data Center Network Infrastructure

Data Center (DC) is constituted by servers interconnected with switches through high-speed communication links within the DC. DC Network (DCN) can be characterized by its physical resources building the network topology i.e., routing/switching equipments and the network protocol deployed to manage the traffic. Also, the interconnection between physical resources such as servers and links can be deployed through different designed topologies.

Besides, thanks to the virtualization techniques, DCN are able to host a high number of servers which can run a massive number of virtual machines. Conventionally, a DCN is formally composed by the following devices [23]:

- *Switches*: are the network equipment responsible of i) the interconnection between hierarchical switches (i.e., core and aggregation switches for traditional DCN) via high-speed communication links (up to 10 Gbps), ii) the inter-racks communication by interconnecting servers within different racks, and iii) the connection between servers within the same rack (i.e., ToR switch placed on the top of rack for traditional DCN). Thus, we note that the switch speed and its number of ingress/egress port can limit the server performance.

- *Servers*: are the main physical component of the DCN. It is responsible of running multiple cloud applications, transmitting and storing data within the DCN.

- *Racks*: are the servers and switch containers that helps to i) facilitate the installation and the communication of DC hardware and ii) optimize the DC space.

- *Cables*: insure the interconnection between servers and switches and can transport electricity or optical signals depending on the required transmission throughput between equipment. Basically, we use the Gigabit Ethernet standard in wired DCN installation.

1.2.2 Data Center Network Architecture

Over the last decade, several research studies addressed the Data Center architecture to deal with scalability issues and the increasing voluminous traffic.

In [24], data center architectures are classified into three categories: i) switch-centric topologies, ii) server-centric topologies and iii) hybrid recursive topologies depending on the role of physical component in the DCN (i.e., switches and servers) for transmitting packets.

In the first category, we cite the traditional DCN i.e., conventional multi-rooted tree-like DCN [23], where switches are hierarchically deployed to transmit the traffic between racks. This

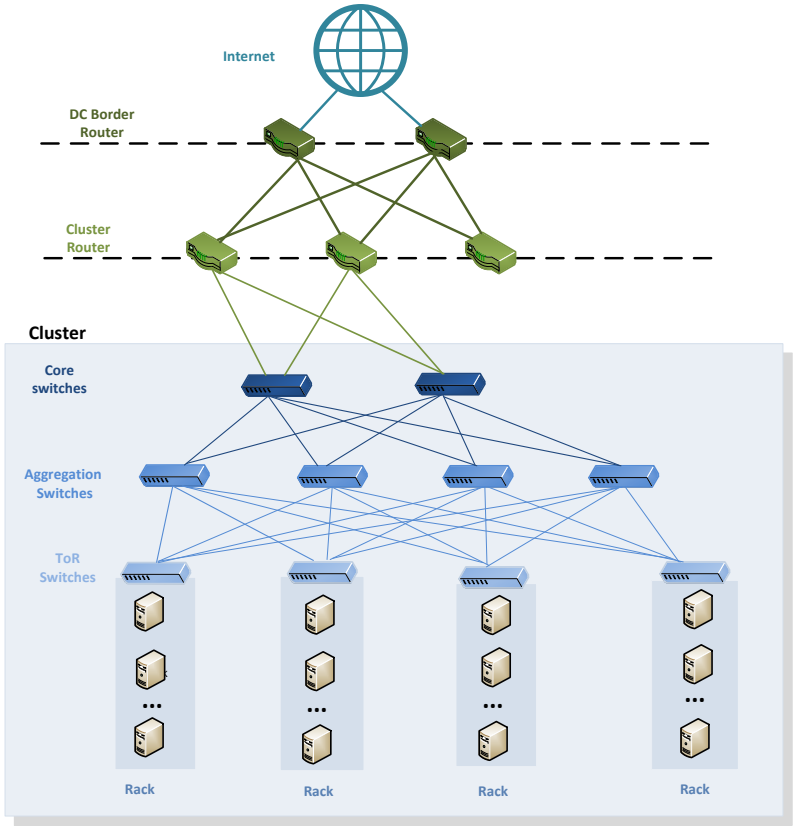

Figure 1.2 – Conventional Multi-tiered DCN Topology

category is considered as a three-layers hierarchical DCN tree. The first layer is composed of core switches which are responsible of the interconnection between aggregation switches. The latter is deployed in the second layer and manage the communication of the edge switches or the ToR switches located on the top of each racks in order to establish the inter-racks communications as it is shown in Figure. 1.2.

To deal with the scalability issues and to alleviate congestion problems, several DCN solutions inspired from the multi-tiered tree topology have been proposed such as: Clos Topology [25], Fat-Tree [26], Portland [27] and VL2 [28]. These data center network topologies are the most commonly deployed (e.g., Facebook [29] and Google [30] data centers deploy and customize the Clos topology to meet their services requirements). However, these topologies still suffer from devices performance limitations which impact the DCN efficiency for managing a continuous growing data center traffic. In this regards, recent approaches addressed new data center designs in order to achieve higher network performance and to get an unprecedented flexible architecture supported by DCN [31]. To do so, Optical DCN (ODCN) and wireless DCN based

on antennas were developed to augment DCN infrastructure and boost traffic management by installing new antennas for inter-rack communications or by adding new highly speed optical links.

In ODCN architectures, optical switches and cables were deployed to enhance the network device communications by installing a high-speed connections. ODCN can be designed through two ways i) fully optical topologies [32][33] [34] and ii) hybrid ODCN (i.e., Optical/Ethernet) [35]. Nevertheless the promising higher-bandwidth and the flexible on-demand links compared to Ethernet cables, the ODCN designs still hard to deploy in real data center environment. In fact, ODCN designs are suffering from the expensive cost to install optical links and the complexity of their deployment in large-scale DCNs [36].

Wireless DCN topologies were proposed to overcome the aforementioned limitations. We distinguish two kinds of topologies based on wireless communication i) fully wireless DCN [37] [38] and ii) hybrid DCN [39]. In these designs, 60 GHz wireless antennas devices are placed in the ToR level for inter-rack communication. Thus, in fully wireless topology, wired links are removed and for hybrid architectures both wired and wireless interfaces are deployed to boost the network performance. In fact, these DCNs architectures provide a high flexibility by harnessing on demand wireless links in a dynamic way which help to alleviate congestion and improve network performance with a high throughput. However, wireless DCNs endure some technical limitations particularly for wireless links which still suffer from noise and interference factors and hence strong repercussion on signal quality for cloud services.

In hybrid recursive topologies, both servers and switches are responsible of packet transmission within DCN. In these architectures, servers perform routing while the mini-switches interconnect a number of hosts. The latter design enhances fault tolerance and ensures a high network expansion by managing millions of installed servers. Furthermore, in these data center infrastructure, we can easily improve data transmission through servers than commodity switches which enhance innovation and design flexibility such as in DCell [40], BCube [41] and FiConn [42] topologies. However, DCell and BCube DCNs can suffer from the overhead and wiring costs due to the high number of NIC ports at end-hosts [43].

For server-centric topologies, only servers are responsible for transmitting packets within the DCN in addition to their traditional role of processing and computing. In this kind of topologies, few studies have proposed data center designs based only on servers to perform both forwarding and processing data. For instance, CamCube [44] is a 3D torus topology using servers along each axis. The latter architecture was inspired from the Content Addressable Network (CAN) overlays [44]. Each server is directly connected to 6 neighbour servers. The main advan-

tages [43] provided by this topology are that: i) CamCube guarantees a robust fault-tolerance despite of 50% of servers link failure, ii) improvement of routing innovation based on the location of server in space to transmit the hashed content (i.e., key-based server-to-server routing [45]) and iii) adaptive application routing techniques. Nevertheless, CamCube topology suffers from a high network diameter which increase the latency and traffic loss in the network.

Despite the several data center solutions proposed in tree-based networks aiming to improve the routing and the network performance, unfortunatly they still suffer from several limitations. Therefore, the Hybrid recursive topologies were proposed in order to overcome the traditional DCNs drawbacks. In fact, the hybrid DCN succeeded to enhance the DCNs performance by allowing servers, in addition to switches, to forward packets. However, these data center architectures still suffer from switches limitations e.g., related to limited switch ports number to deal with tremendous traffic for large-scale topologies. Thus, Server-centric topology seems to be an alternative solution to deal with the previous weaknesses. Particularly, the latter design can provide an adapting routing for running applications while considering the data center network expansion. Hence, in this thesis, we assume CamCube data center topology to optimize the routing and the resource allocation within it. Nevertheless, CamCube DCNs suffer from a long routing diameter which may induce a negative impact on the QoS network performance. Consequently, in this thesis, to overcome the CamCube drawback, we propose a new routing scheme based on SDN architecture. Our addressed schemes tackle the routing and resource allocation for both online (i.e., unicast and multicast) and batch servers communication mode. Our propounded algorithms aim to improve simultaneously the CamCube network expansion and its Quality of Service in terms of latency, packet loss, path length and jitter.

It is noteworthy to say that despite of the aforementioned infrastructures constraints, these topologies were meticulously designed to provide the best solution in order to meet customers growing needs. To do so, researchers had to deal with several technical and cost challenges. Hereafter, we will present the main challenges that DCN designers have to deal with in order to outfit the users requests.

1.3 Data Center Challenges

The emergence of cloud computing and web applications incite warehouse-scale data center operators to require much-higher QoS performance in order to sustain the increasing network traffic. Moreover, new features should be installed in these fabrics in order to fulfill the increasing complexity and sophistication of data center applications. In fact, the proposed DCN de-

signs are suffering from several drawbacks that prohibit them to meet the current voluminous demand of online services.

In this section, we enumerate the most relevant challenges to take into consideration in the design data centers:

- ***Scalability***
  The main objective of deploying DCN is the ability of running a huge number of services applications through an extended infrastructure where companies can centralize their services. Besides, in order to improve provider profit, the efficient and effective expansion represents a key challenge. Currently, the continuous growing demands can have a negative impact on the performance and the scalability of the network [46]. So, several companies such as Facebook focuses their studies on this field regarding the deeply induced impact that it can have on their businesses [47]. The company aims to extend its data center infrastructure by adding 30000 m^2 sized building in 2023 in Odense [48]. In fact, once a topology upgrade is realized, the whole network should be expanded through a meticulous design plan in order to i) avoid fragmentation, ii) allow an efficient utilization and iii) face a trade-off between the scalability challenges and added equipment cost. Indeed, some DCN architectures such as hierarchical designs may suffer from high construction costs since they need to increase the number of highly performed Ethernet switches to scale [49]. Similarly, optical DCN make use of optical interconnection networks that are not only expensive to deploy but also present a technical challenge when installing central MEMS Switches with limited capacities to enhance their capabilities of hosting hundreds of thousands of servers [50]. To overcome these challenges, the authors in [51, 52, 53, 54] introduce some proposals that aim to enlarge DCNs while keeping the existing hardware legacy. Indeed, these studies intent to maximize the bisection bandwidth[1] and the data center reliability [43]. Nevertheless, they often propose high expectations like proposing `LEGUP`[2] [52] for tree-based network or requiring new routing mechanism [53].

- ***Resilience and Fault tolerance***
  It is considered as a crucial concern in data center networks. Actually, these fabrics can suffer from failure engendered by the hardware, software or the power outage problems of physical equipments (i.e., servers, switches, links and racks). Indeed, to ensure the fault tolerance, both redundancy in physical connectivity level and solid protocol mechanisms

[1] The bisection bandwidth is equal to the minimum of bandwidth required of all links capacities within a bisection. The latter is considered as a partition of an equally-sized number of nodes within the network [55].

[2] `LEGUP` was proposed in tree-based DC in order to provide a new design and physical arrangement for upgraded and expanded topologies. The proposal provides a high bisection bandwidth for an upgraded network while reducing its cost to the half.

designs should be deployed. In fact, in some DCN designs e.g., traditional and hierarchical topologies, each aggregate switch is connected to a group of core switches not only for enhancing the connectivity and the inter-equipment communication but also to avoid the link failure [24], hence the resilience of the architecture. Similarly, the number of links between ToR and aggregate switches is augmented in both hierarchical and server only topologies. In the latter topology, a multi-network interfaces server can reach other ones through several possible links. Therefore, end-to-end communication and installation of highly-performed links between DCN equipment should be considered when designing data centers in order to provide the needed services to customers with the requested QoS by minimizing the latency and packet loss rate. Besides, to avoid the data center failure, authors [56] address the improvement of the data centers resilience by creating systems software to support this redundancy between data center which will reduce the cost engendered by the increased deployment of physical equipments. Moreover, several studies [57][58][40] addressed in their proposals the fault tolerance and resilience challenges when designing the proposed DCN architectures.

- ***Congestion control***

  Some cloud and big data applications can actually requests simultaneously a high number of synchronized and interconnected servers to communicate with a single host e.g., Map Reduce of Hadoop and search applications. This mechanism can rapidly create a bottleneck in the server which can be responsible of a severe degradation of data center QoS by increasing the response time or the drop of transmitted packets (caused by an overflowed switch buffer). Consequently, the application throughput can immediately decrease as result of packet loss and TCP Retransmission TimeOut (RTO). The latter can engender up to 90% of application throughput deterioration [49]. Indeed, a congested data center network can cause the deterioration of cloud services for end users. Hence, given the importance of its impact, authors in [59, 60, 61, 62] propose new alternative solutions to overcome this challenge by addressing new congestion control algorithms. For instance, the authors in [59] propose the Incast Congestion Control for TCP (`ICTCP`) scheme that adjusts in advance the TCP of the received window before a packet loss occurs. The proposed design aims to minimize the impact of the Incast Congestion by decreasing the timeout value. Also, in [60], the authors propose the Quantized Congestion Notification algorithm (`QCN`) that has been addressed to ensure the Ethernet congestion control for hardware implementation. Moreover, the authors in [62] proposed a scalable, light-weight and flexible algorithm for data center congestion control able to manage non-conforming flows called `DCTCP`. The proposed scheme makes use of fine-grained control over random TCP. Hence, the propounded solutions aim to alleviate the congestion within data center networks by

enhancing the QoS provided by its tenant applications.

- ***Load balancing and network QoS optimization***
  Load balancing in DCN aims to uniformly distribute the workload among DCN elements by forwarding flows through multiple paths. In fact, cloud-oriented DCN generally provides path diversity to manage horizontal scaling for unpredictable arrived traffic from numerous applications. Some DCN topologies such as Fat-Tree [58] and Clos [25] make use of dense multi-path architectures to support a large bandwidth for internal data exchange [49]. Then, it is compulsory to use a balanced workload for efficiently utilizing the network resources. For intra-data center networks, the traditional topologies uses the Open Shortest Path First i.e., OSPF for servers communication where the shortest path is calculated in advance regardless load balancing through multiple path [49]. Besides, to deal with load balancing challenge, data center topologies can deploy i) uniform and ii) non-uniform multiple path for servers communication within their networks [43]. The authors, in [63], investigated the Equal Cost Multi-Path (`ECMP`) used for splitting traffic through uniform multiple paths[3] routing. However, `ECMP` is unable to use the overall available capacity in these multiple routes [64] and can engender congestion since it does not check the load before assigning paths [65]. On the other hand, non-uniform multi-path, need to take more factors into account to run its mechanisms such as: path latency and current link load. Several proposals [66] [67] investigated this challenge. However, the proposed solutions were not able to reach the desired response time such as for the flow scheduling system namely `Hedera` [68], or proposed a solution dedicated for a particular data center topology such as the Port-switching based Source Routing (`PSSR`) for SecondNet [66]. Besides, in [69] the authors proposed an enhancement of `ECMP` performance. Different proprieties of the cloud applications make the load balancing more complex to establish within DCNs. Therefore, more studies and investigations still be needed to address both uniform and non-uniform multiple path flows scheduling for the different data center designs in order to propose innovative protocols and mechanisms that better exploit the available hardware capacities.

In this thesis, we consider the scalability, load balancing and the optimization of QoS Network challenges. To cope with the latter issues, we describe hereafter the problem statement for optimizing both routing and resource allocation within SDN-based CamCube topology.

[3] Uniform path is an equal-cost path[43].

1.4 Problem Statement

In this thesis, we tackle the challenge of packet routing and the resource allocation problem within CamCube topology. Particularly, we address unicast and multicast communications for online mode. Afterwards, we consider the arrival of traffic flows in batch mode. As a consequence, we need to provide efficient mechanism in order to optimize the overall resource allocation within CamCube DCN. So, we require to propose new algorithm schemes that are able to i) efficiently manage the resource allocation and ii) optimize path between a source and its destination(s) for both online and batch communication modes in order to enhance the CamCube network performance while respecting the QoS constraints defined by the cloud customers. For these purposes, our proposals will take into consideration:

- physical and virtual hardware (e.g., virtual switches) limitations such as defining the number of Open Virtual Switch (OVS) ports responsible for forwarding packets between servers and number of maximum allowed packets in the switch waiting in the queue in order to minimize the packet loss.
- the end-to-end transmission delay and rejection of the arrived data flows.
- the congestion issues within the network equipment by balancing the load and exploiting residual available CamCube links capacity.

The above constraints affirm the complexity of the routing and resource allocation problems in a dense topology as CamCube DCN. For that reason, we make use of the mixed integer linear programming formulations and the combinatorial optimization to reach the best (optimal if it is possible) solutions.

Moreover, DCN are composed of different type of networks such as Storage Area network (SAN) or High Performance Computing (HPC) network, and Local Area Network (LAN), etc. All these networks require a high network availability to efficiently run services and provide the needed data despite of the voluminous customers requests. These performance requirements endorses the need of centralizing DCN management via an SDN controller in order to i) guarantee a high throughput and loss-free deliver, ii) avoid congestion owing to routing conflicts (same switches ensuring multiple flows to different stations meantime), and iii) over-utilization of links [70].

Current SDN solutions are able to manage the DCN by using APIs that retrieve real-time information about the topology and network equipment state. In this manner, we need to synchronize the proposed algorithm with the real-time network information in order to:

- get the state of the network (exploiting available links and calculate the current residual bandwidth in the CamCube links)
- install the flow via the OVS existing module managed by the SDN controller following the obtained path solution.

Consequently, to deal with these aforementioned challenges, we propose to deploy ONOS SDN controller providing a very performed and customizable modules and/or APIs.

1.5 Thesis Contributions

In this section, we summarize our main contributions of this thesis where we address the routing and the resource allocation challenges for SDN based CamCube topology detailed in Section 1.4.

SDN based CamCube Topology
In this work, we implement and evaluate CamCube server-only DCN topology based on the shortest path routing. To do so, first we make use of Mininet to build the emulated CamCube DCN network. Then, we deploy ONOS [71] SDN controller with the emulated CamCube by integrating the south-bound protocol OpenFlow and the north-bound APIs with the routing application layer. Hence, SDN ONOS controller is able to: i) install the flow rules in CamCube DCN and ii) get a real time information of the full topology and the state of network (e.g., residual bandwidth, error rate, latency, etc.). Afterwards, we perform the shortest path routing protocol (i.e., OSPF) and analyze the results with our emulated CamCube DCN environment (Mininet / ONOS) in terms of packet loss, jitter, and latency.

Unicast routing protocol in SDN based CamCube topology
In this contribution, we aim to propound a new algorithm to treat routing and resource allocation problem within SDN based CamCube topology. We only focus in this chapter on the unicast communication between servers. Our objective is to minimize the end-to-end delay and the packet loss while maximizing the residual bandwidth in the CamCube DCN. The main idea consists on proposing a new scheme providing an optimal path between a randomly designed source and destination within CamCube servers. To deal with these challenges, we propose the Camcube Routing Protocol `CRP` to address the routing problem. `CRP` aims to minimize the total throughput in CamCube DCN and the number of traveled hops of the proposed optimized path. The problem is formulated as Mixed Integer Linear Programming (MILP) model to optimize and we analyze the efficiency of our proposal through extensive experimentations using Mininet emulator and ONOS SDN controller. Afterwards, we noticed that the proposed `CRP` needs long

duration to converge. Consequently, we proposed a new algorithm (ACO-CRP) based on meta-heuristic Ant colony optimization in order to reduce the convergence time. Later, we compare our proposals to the shortest path protocol in both CamCube and Clos toplogies. Finally, we conclude that the obtained results outperform the prominent related strategies in terms of latency, packet loss, jitter, E2E delay and path length.

M-CRP routing protocol in SDN based CamCube topology

The majority of the distributed and cooperative applications are simultaneously communicating with each other. Indeed, multicast routing is considered as a solution to avoid the congestion DCN. To do that, we emulate the CamCube DCN managed by the ONOS SDN controller to optimize the forwarding of multicast flows. For this purpose, we first formulate the multicast routing problem as a lexicographic multi-objective optimization. Our objectives are to maximize the residual bandwidth and to minimize the number of relay nodes in the multicast tree. Then, we propose new SDN application named Multicast CamCube Routing Protocol (M-CRP). It re-formulates the problem as a single-objective ILP. M-CRP is based on Branch and Cut algorithm and takes into consideration the state of CamCube DCN thanks to the real-time monitoring performed with ONOS controller over the OpenFlow Southbound protocol. Next, we deploy our application within our emulated CamCube DCN controlled by ONOS. However, preliminary results show that M-CRP is very bad in term of convergence time. Consequently, we proposed a new meta-heuristic algorithm based on Ant Colony Optimization (ACO-MCRP) to overcome the convergence time problem. Finally, based on extensive experimentations, we compare M-CRP and ACO-MCRP with the shortest-path approach in Camcube DCN and OSPF in Clos DCN. The QoS results obtained are very satisfying in terms of packet loss, latency, and jitter.

Batch-CRP routing protocol in SDN based CamCube topology

The obtained results of the aforementioned proposed protocols show that ACO-CRP and ACO-MCRP outperform the traditional unicast protocol based on shortest path in terms of packet loss, latency and jitter. However, this flow treatment needs to request the SDN controller for each arrived flow in the network forwarding elements. So, to manage a huge intra-data center traffic volume and to handle scalability issues, this high frequency of SDN solicitation can lead to performance deterioration. It can also engender the flow loss especially in a congested network. So, the idea of treating arrived flows through **batch routing scheme** can be an alternative solution to overcome these limitations. Indeed, the SDN controller can simultaneously treat a number of flows by calibrating some batch window size. In doing so, we will decrease decision process duration of flows while enhancing the system scalability and QoS performance.

In this regard, in this contribution, we start by formulating the batch routing problem as a lexicographic optimization multi-objective. By this proposal, we aim to maximize the number of

accepted flow in our system, to maximize the residual bandwidth, and to minimize the traveled hops to reach the destinations. Afterwards, we propose a new multi-phase batch routing algorithm in CamCube topology denoted by batch-(M)CRP based on SDN application to treat unicast and multicast flows. Indeed, we define batch-CRP protocol to address unicast flows and batch-MCRP to treat multicast flows. We reformulate the problem as a multi-single objective problem in aim to make use of branch and cut algorithm to solve it. Our batch-(M)CRP retrieves the real-time network state of CamCube DCN using ONOS controller via its Openflow southbound interface. Then, we emulate the CamCube topology managed by ONOS controller and study the performance of our proposal through extensive experimentations by comparing it with the shortest path in both CamCube and Clos DCN. The obtained results show that batch-(M)CRP achieves good performances in terms of packet loss, latency, and jitter.

1.6 Thesis Outline

The remainder of this thesis manuscript is organized as follows. First, in Chapter 2, we will overview the different data center topologies found in the literature. Then, in Chapter 3, we will overview the related routing protocols within the different DCN topologies. In Chapter 4, we will discuss the management of CamCube DCN by the SDN controller. Next, in Chapter 5, we will detail the proposed CRP and ACO-CRP unicast routing schemes for CamCube topology. Later, in Chapter 6, we will describe the M-CRP and ACO-MCRP protocols designed for CamCube topology. Chapter 7, will present the batch-(M)CRP routing scheme proposed for CamCube data center. Afterwards, Chapter 8 will concludes this manuscript and will provide some insights of some future work and perspectives. Finally, we will present the different tasks elaborated in Devoteam Innovation team during the PhD thesis in the Appendix.

Chapter 2

Taxonomy of Data centers Designs

Contents

2.1 Introduction

Due to the widespread use of cloud-based services and the ever expanding traffic within data center networks, the scale of this latter is continuously increasing. Therefore, data center network infrastructure should be more agile and re-configurable in order to follow the rapid change of cloud applications demands and their services requirements. Hence, significant research work are focusing their studies on designing a high performed data center architectures.

In this chapter, we will study in-depth the DCNs designs by providing a taxonomy of the most relevant data centers architectures. So, we will present a detailed analysis of data center networking infrastructure. Afterwards, we will summarize all the enumerated solutions by making a qualitative comparison based on several performance metrics and DCNs properties.

2.2 Data Center Architecture

In this section, we provide a classification of existing DCN architectures and we detail the designs conception of each cited class. Generally, providers take into account several criteria to build their data center infrastructure such as scalability, fault tolerance[1], bandwidth availability, easy installation, agility, etc. Actually, we discern two main research approaches. For the first direction, scientists built wired DCN architectures and upgrade them in order to improve scalability and to minimize cost effectiveness. For the second direction, researchers propose to implement and set up new networking techniques aiming to improve data centers performance and to deal with previous architecture challenges. Finally, we provide a new classification of these proposed solutions.

2.2.1 Classification of Data Center Infrastructures

Figure. 2.1 provides a taxonomy of DCNs architectures through two main classes: i) Typical data center topology based on wired cabling and ii) Enhanced data center designs based on hybrid (wired and/or wireless, and optical) cabling techniques. Hereafter, we will detail the variety of categories in these two groups. Note that this figure provides a glimpse of the proposed DCNs designs examples for each class. In the first category, DCNs architectures are cabled only through wired connections to ensure the servers communication within it. Authors [24] distinguish three main groups of DCNs designs depending on the role of network/physical equipment for transmitting packets within data center networks as follow:

[1] Fault-tolerance allows a defined architecture to continue to operate properly after an appeared event failure on its equipment e.g., switch link failure.

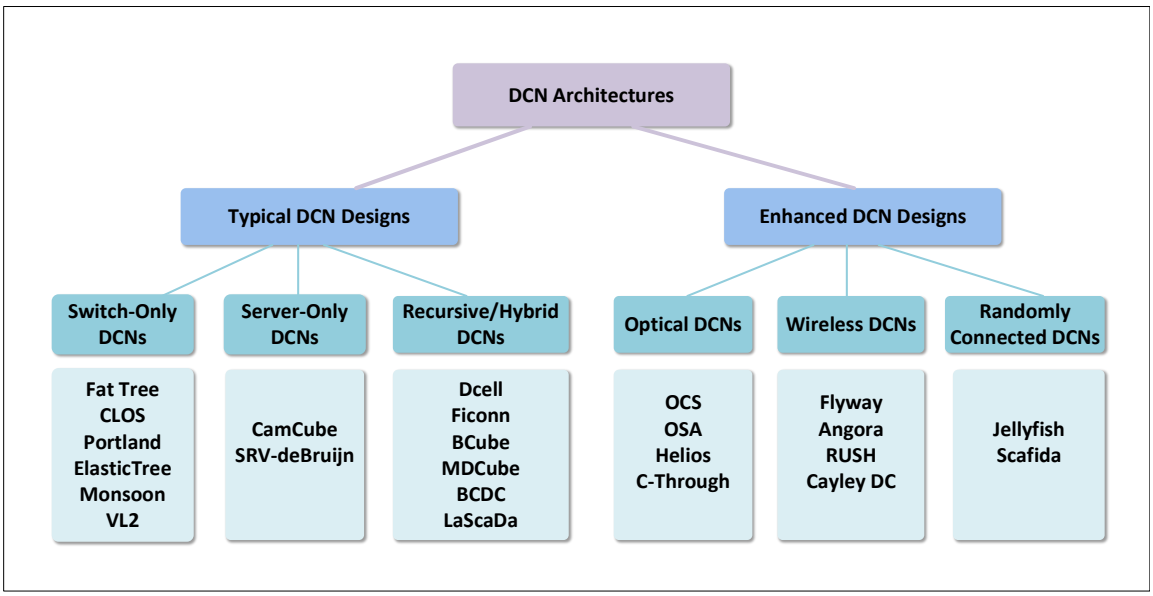

Figure 2.1 – Classification of DCNs Topologies

- ***Switch-only designs:*** In these topologies, only switches are responsible of the functions related to network i.e., transmitting packets between servers for intra and/or inter-rack communications. These designs aim to improve the network scalability, to minimize the over-subscription ratio[2] and to accelerate the flow transmission. We can define three main classes for switch-centric architectures related to their structured characteristics: i) tree-based DCNs where switches communicate with each other through a multi-rooted form. ii) Hierarchical switch-only DCN where switches are linked through several layers and each layer variously manages the traffic. iii) Flat DCN where switch layers are compressed in only or two layers in order to facilitate the traffic management and the DCN maintenance.

- ***Server-only designs:*** In this category, only servers are responsible for both transmitting packets and running applications. The main goal of these architectures is to improve DCNs QoS and to outfit the cloud applications requirements by providing adaptive routing techniques and a highly-scalable networks.

- ***Hybrid recursive designs:*** For these architectures, both switches and servers have network functions. In fact, servers are responsible of improving the management of network

[2]The over-subscription ratio is used to closely monitoring the aggregated bandwidth of Core and Access switches ports in the DCN. It is calculated as a ratio between the upstream and the downstream capacities of linked switches to well manage the traffic pattern within DCN.

functions. Indeed, servers not only can play a role of an end-hop but also a relaying-node for multi-hop data exchange. However, switches keeps their traditional role i.e., responsible of forwarding packets within the DCN networks. Generally, these topologies are constructed through a recursive multi-layer topology in which each layer presents a level in the DCN architecture.

For the second category, Enhanced topologies were proposed for the incoming Cloud computing services. In fact, to overcome wired DCN limitations, the enhanced DCNs deploy new networking techniques mixing wired and wireless strategies or replacing Ethernet cable with faster-transmission optical link. Hereafter, a summary of the main proposed classes of enhanced DCNs topologies.

- ***Optical DCNs:*** In these designs, cloud architects make use of optical devices to enhance the transmission speed. The optical DCNs can be: i) full optical where only optical devices are deployed within it or ii) Hybrid Optical DCNs where both optical and Ethernet switches are used to transmit packets between servers within the data center network.
- ***Wireless DCNs:*** These architectures use wireless equipments to improve the data center performance. We can distinguish two main wireless DCN architectures : i) full wireless DCN where only wireless equipments are used to connect DCN devices and ii) Hybrid DCN where both wireless and wired devices.
- ***Randomly connected:*** These DCNs are considered as a random architecture. In fact, each switch and/or server can randomly be connected to other switch according to adapted predefined algorithms [53].

2.2.2 Typical DCNs design: Overview

2.2.2.1 Switch-only Tree-based DCNs

The traditional switch-only DCNs are typically based on three switch layers forming a multi-rooted tree-based architecture as illustrated in Figure. 2.2. The latter design is characterized by: i) the core layer represented at the top level, ii) the aggregation layer is situated in the middle level, and iii) the edge layer directly connected to the data center's racks. We make use of the core layer to connect the data center to Internet due to the high-speed up-links characterizing the core switches located in this level. Nevertheless, the aggregation switches provide 10 $Gbps$ links to the ToR switches i.e., edge switches. The latter provide only 1 $Gbps$ links to interconnect servers of the same racks. However, the multi-tiered DCNs suffer from several technical limitations impacting their performance. In fact, because of the growing traffic demands, the

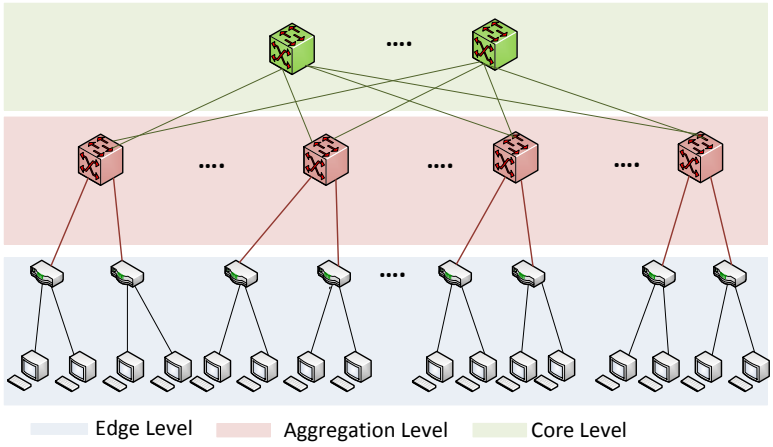

Figure 2.2 – Traditional Tree based DCN Topology

workloads can reach the peak and cause bottlenecked core switches. Also, these DCNs architecture are based on switches for traffic transmission. So, they only depends on switch ports to scale which can increase their costs and energy consumption. Besides, traditional DCNs encounter resiliency issues. Indeed, servers communication can be interrupted when aggregation switches links fails. Moreover, this kind of design suffers from traffic engineering issues related to an unbalanced resource allocation within it. Consequently, several alternatives were proposed by researchers to deal with these limitations.

To overcome the aforementioned challenges, we can cite the most relevant hierarchical proposed topologies such as `Clos` [49], `Fat-Tree` [58], `Portland` [27], `ElasticTree` [72], `Monsoon` [73] and `VL2` [28] data centers. In fact, despite of their multi-layered architecture, these solutions aim to minimize the congestion and to reduce the over-subscription in the inferior layer switches depending on the superior level switches.

In 1953, Charles Clos proposed the mathematical theory of `Clos` topology which is considered as an enhancement of the architecture based on Tree [49]. This topology was designed to be a multi-stage and non-blocking architecture aiming to offer higher bandwidth than the frequency range supplied by a single switch. As shown in Figure 2.3, `Clos` is a multistage switching architecture where each level is connected to all units of the lower level in order to minimize the intersecting nodes when the input and output streaming is augmenting. `Clos` topology is characterized by the leaf layer which is directly connected to server and then composed by ToR switches. It defines the over-subscription ratios which determine the number of spine switches in the core layer. The latter is responsible of connecting all the ToR switches through the edge aggregation composed by the aggregation switches. `Clos` architecture prevents the transition

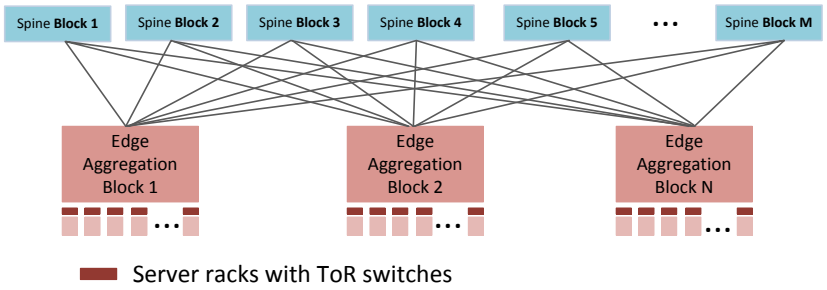

Figure 2.3 – Clos DCN Topology

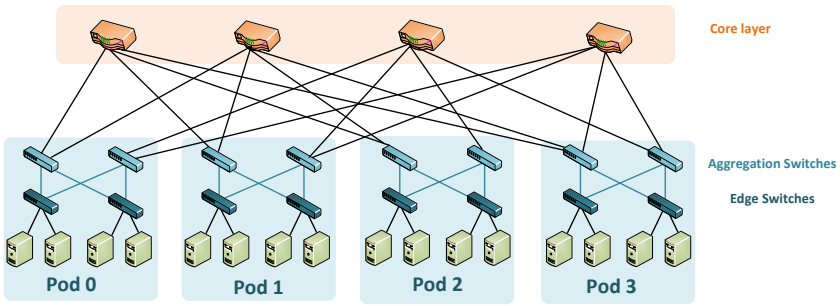

Figure 2.4 – Sample of Fat Tree Topology

of traffic through horizontal links existing in the same layer. This fabric is implemented for small and medium network scalability. Also, `Clos` topology can deploy a limited number of switches while enhancing the number of ports to establish huge connections. In fact, the increased number of switch ports can enlarge the spine layer width which can minimize the congestion when scaling up. Moreover, this topology is characterized by an important link redundancy where each leaf switch is attached to all spine switches in order to enable the multi-path routing and avoiding the over-subscription engendered by OSPF routing protocol through its conventional link state. To conclude, `Clos` network provides a multiple paths to avoid communication block within Clos switches and a better scalability in comparison with the traditional multitiered DCNs. This architecture improves the bandwidth utilization within its network particularly in aggregation level. However, this design still involves a high number of links and homogeneous switches which can impact on the cost.

A particular instance of `Clos` architecture is `Fat-Tree` DCN. It was proposed by the authors in [58] and illustrated in Figure. 2.4. This design is leveraging predominantly commodity Ethernet switches to tolerate the full aggregate bandwidth of racks hosting tens of thousands of servers. `Fat-Tree` proposed a new switches interconnection design based on *pods*. In fact, we

suppose a Fat-Tree based on n *pods*. In each Fat tree based cell i.e., *pod*, we can connect $n/2$ edge switches to $n/2$ aggregation switches and the remaining $n/2$ edge switches ports are connected to $n/2$ servers. Also, there are $(n/2)^2$ n-ports switches attached to the n *pods* and form the core level of the tree. Consequently, a Fat-Tree topology deploying n-port switches can support $n^3/4$ servers. The proposed topology minimizes the cost by deploying identical cheap commodity switches. Moreover, the important path diversity existing in each level allow a non-blocking communication between servers and can deeply minimize the congestion effects particularly for a communication between a given couple of ToR switches. However, this topology can suffer from scalability issues especially when it depends on the number of switch port for network extension. Hence, it can also allow complicate connections and eventual lower layers device failure which can negatively impact on the topology performance for a large-scale design.

Lately, several topology designs have been proposed to overcome Fat-Tree topology challenges and to provide better performance to its users. The most relevant of them are: Portland and ElasticTree.

Indeed, inspired from the Fat-Tree topology, the authors, in [27], established the Portand design. It is considered as a scalable and a fault-tolerant two-level routing architecture. Portand consists on deploying a fabric manager and Pseudo MAC (PMAC) address to transmit data packets then a MAC to PMAC mapping to prevent modification within servers. Each pod in Portland topology is characterized by a unique number and position learned by the leaf i.e., edge switch. These identifiers are assigned through an employed discovery protocol. The edge switches appoint a $48-bit$ PMAC to a server in order to identify i) the port number connected to the server, ii) the number of VM deployed in the physical machine and iii) the corresponding pod's identifiers. Hence, this PMAC, existing in the fabric manager, allows a host to communicate with other servers within the Portland network. Also, the fabric manager ensures the Virtual Machine (VM) migration between hosts, maintain the VMs information and communicate the updated events to its previous physical host. In this regards, Portland architecture built from Fat-Tree design, proposed a new two-layer routing scheme aiming to enhance the fault tolerant routing, the network scalability and VM migration. Nonetheless, Portland topology presents some weaknesses. In fact, the deployed Portland switches need an adaptive set up and modifications to provide these services. Also, Portland topology can suffer from devices failure as its predecessor since it is based on the fabric manager to insure the aforementioned functions.

Furthermore, in 2010, the authors in [72] proposed ElasticTree as an enhancement of Fat-Tree topology. The main feature of this architecture, illustrated in Figure. 2.5, is to provide

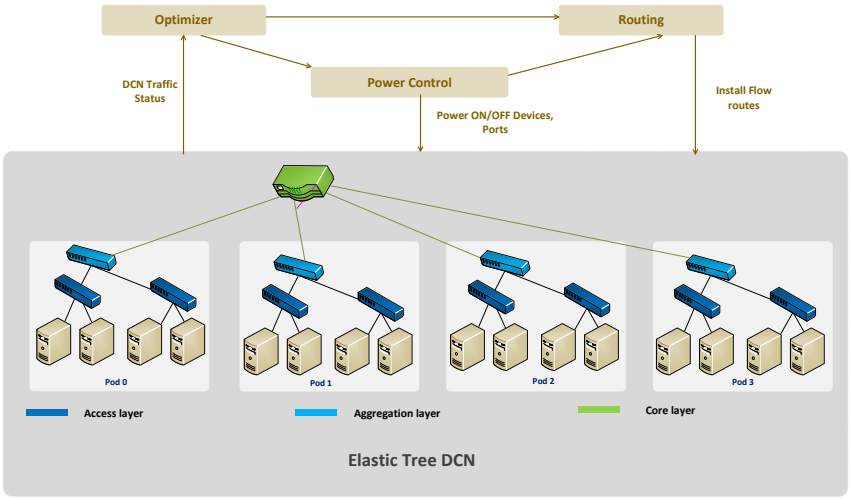

Figure 2.5 – Sample of Elastic Tree Network Architecture

a flexible use of switch i.e., enabling or not connection on demand. So, the main idea of implementing such a design is to save energy consumption by affording continuously full bandwidth across all edge switches. Hence, three principal logic modules compose this design: i) *optimizer* to only enable equipment that can provide requested bandwidth according to a data flow conditions, ii) *routing* to provide the route of the data flow, and iii) *power control* responsible of managing the state of required equipment for routing the data flow e.g., ports, adapters and switches. The main advantage of this design is to save energy consumption and then reducing the maintenance cost of the data center. Nevertheless, since `ElasticTree` is based on `Fat-Tree` topology, the new proposed design still suffers from the same weaknesses as its predecessor especially for scalability issues and low-layer device failure.
In 1990, the `Valiant Load Balancing (VLB)` scheme [74] was proposed to deal with the variation of traffic in processor interconnection networks. These designs aim to mitigate hotspots for a randomly transmitted traffic through multiple paths. The most relevant architectures belonging to the `VLB` designs are `Monsoon` and `VL2` architecture.

In 2008, Microsoft researchers proposed the two-layered `Monsoon` architecture [73]. This architecture connects a huge number of servers without over-subscription. It consists of i) *access router* located in the second layer and ii) the *core border routers* located in layer 3 and transmit packet by using Equal-Cost Multiple-Path (ECMP) through multiple path and the VLB mechanism to ensure the load balancing. Despite of the aforementioned enhancement related to the over-subscription, the `Monsoon` design is considered as incompatible with data centers based on Ethernet connections since it changes the typical Address Resolution Protocol (ARP) and

proposes a new MAC interface to transmit encrypted Ethernet frames.

Lately, the authors in [28] introduced the VL2 architecture based on virtual two-layer Ethernet to connect servers located in the same LAN contrarily to Fat-Tree and aims to dynamically allocate resource within its network. VL2 design is considered as a tree-based topology that increases connection by using the conception of Clos topology and the VLB mechanism to ensure the load balancing of routing within the DCN. In addition, VL2 uses the ECMP protocol like Monsoon to route packet and transmit data along optimal multiple paths and manage VM migration by resolving the address redistribution problem. Nevertheless, the advantage that presents VL2 topology, the latter still suffers from the same weaknesses as most of the tree based architectures such as scalability and single node failure impacting reliability.

Moreover, the Flat DCNs were proposed in order to reduce the multiple switch layer into one or two layers for deployment and maintenance simplicity. To do so, the authors in [75] introduced the FBFLY architecture. It is an energy-efficient data center aiming to balance the traffic load with the energy consumption. Hence, FBFLY adopts the links capacity to the requested traveled traffic by supplying several limited-capacity links instead of 40 Gbps ones. Afterwards, C-FBFLY [76][77] design was proposed as an improvement of the previously mentioned DCN by preserving the same control plane and deploying optical links instead of wired ones to alleviate installation complexity. Besides, FlatNet DCN [78][79] succeeded to decrease the number of implemented links and switches by $1/3$ in comparison with a classical Fat-Tree topology while ensuring the same performance. In fact, to do so, FlatNet topology deploys only two layers where the first 1-layer is only composed by one k-port switch connecting k servers while the second layer is recursively constructed by k^2 1-layer of FlatNet topology. The main advantages of this design is the fault-tolerance based on its two-layered architecture and the use of performed routing protocol ensuring the load balancing.

To conclude, tree based switch only data center provides an efficient multi-routing schemes, a balanced traffic load and a robust resiliency. However, regardless of all this advantages, these designs still suffer from the single point failure due to its centralized manager caused by the tremendous cloud traffic. Besides, lower switches are less secured and have a limited fault tolerance compared to high-level switches. Moreover, these infrastructures present limited network expansion due to its cabling complexity to built multi-layered switches which can impact its scalability.

2.2.2.2 Server-only DCNs

In this section, we present the most relevant server-only topologies, namely, `CamCube` and `SRV-deBruijn` DCN architectures.

- `CamCube` [44] data center was proposed by Microsoft Researchers in 2010. This topology is a 3D torus (i.e., K-ary 3-cube) only composed of servers, where each node is connected to x other servers as illustrated in the CamCube of dimension $3 \times 3 \times 3$ (i.e., 27 servers) depicted in Figure. 2.6 with $3 \leq x \leq 6$.
  The CamCube server has multi-core processors and is characterized by high performed Network Interface Cards (NICs) with multiple ports [24] to ensure the multiple connections with its neighbors. Each server in this design is identified by its 3D position i.e., (x,y,z) coordinate and operate through a specified CamCube functionality describing how to send and receive packets through one-hop communications. In fact, someone may wonder if the servers could perform both routing and computing at the same time. Indeed, `CamCube` design provides a flexible platform via several APIs in which developers can implement their own routing algorithm on CamCube servers according to their requirement. In this regard, a CamCubeOS operating system was designed for these servers [45] which shows, through experimentation study, a very good performance in terms of computing and networking tasks. In particular, the forwarding processes have a negligible impact on the main task (i.e., computing) and hence the interest of such topology. In the origin design of CamCube, the routing function is a key-based server-to-server routing depending on the position of servers in the infrastructure and operates within a distributed control plan that makes use of a link state protocol (e.g., OSPF) and that may exploit the power of the available multi-paths [44] for multi-sessions transport communications between a given source and its destination. The latter issue can be resolved thanks to Equally Cost Multi-Paths protocol (ECMP).
  Consequently, this adaptive deployment strategie mitigates the additional network performance overhead and demonstrates the efficiency of this infrastructure [49].
  The main advantage of this topology is that it offers high link redundancy i.e., high degree of multiple paths. Hence, `CamCube` is resilient to both server and link failures in comparison with traditional tree structures which consequently alleviate the bandwidth bottleneck within its equipment. Besides, thanks to the adaptive routing that `CamCube` topology offers to the running applications within it, this design provides a better performance in contrast with previous DCN structures. However, this topology often suffers from a long routing path with a high network diameter which can impact its performance and cost.

- Moreover, the server-only DCN based on deBruijn graph, `SRV-deBruijn` [80], was pro-

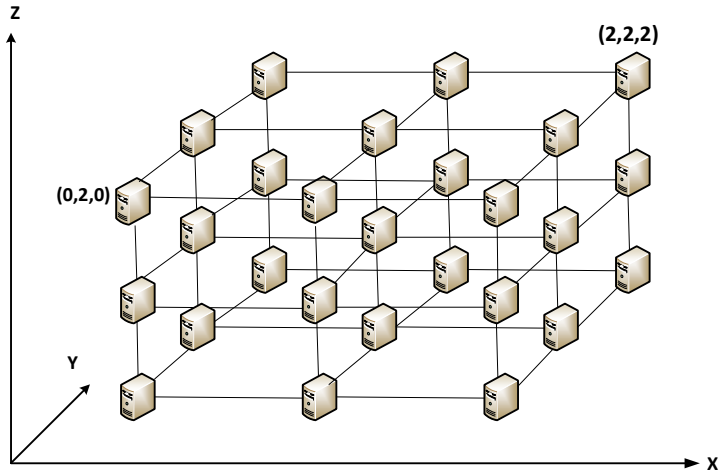

Figure 2.6 – Sample of CamCube Topology

posed to overcome the cost issues engendered by the length of routing path between two communicating servers. In fact, this design arranges labeled servers connecting with their neighbors through a deBruijn graph for intra-rack connections and for inter-racks communications, particularly when servers have the same labels. This design offers a considerable cost-effectiveness and better routing performance comparing to the `CamCube` since it presents only a diameter equals to $\ln(N)$ compared to $\sqrt[3]{N}$.

In this thesis, we consider `CamCube` topology and we propose new routing protocols to overcome the existing challenges in this topology. Hence, we aim by our proposals to enhance the routing and the QoS performance of `CamCube` applications within a high scaled infrastructure.

2.2.2.3 Hybrid Recursive DCNs

Despite of the several proposed tree-based architectures and the different addressed weaknesses, these traditional DCNs still suffer from a high cabling complexity related to the only use of the commodity switches engendering the single point failure through these layers. Therefore, the recursive hybrid DCNs were proposed to overcome this issues and new designs are propounded to enhance the network performance and scalability such as `DCell`, `Ficonn`, `BCube` and its derivative `MDCube` topologies.

`DCell` [40] is a recursive DCN architecture where both servers and switches can relay packets within nodes. `DCell` design is based on units i.e., $DCell_0$, formed by servers and a mini-switch

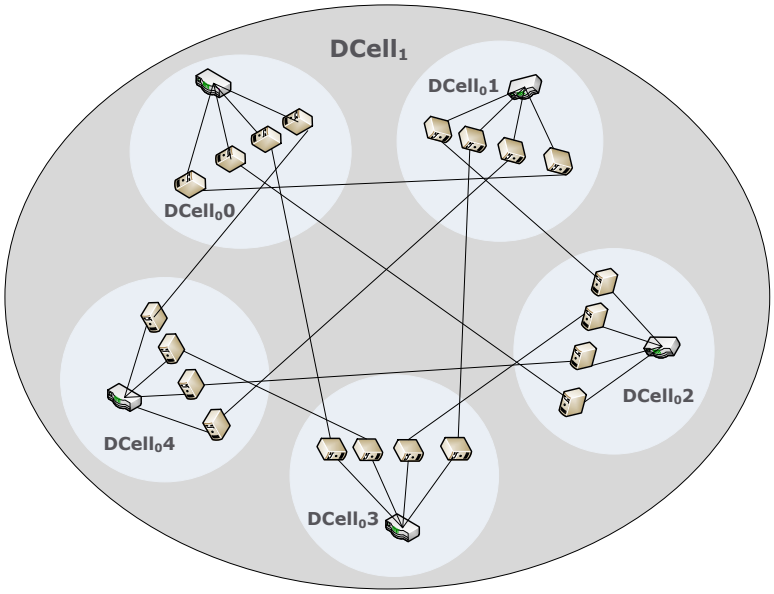

Figure 2.7 – Two-levels DCell DCN Topology

and interconnected with each other to build the whole infrastructure as depicted in the Figure. 2.7. n servers located in $DCell_m$ network interconnected via a n-port commodity switch form the $level_0$ of the topology. The total of $n+1$ $DCell_m$ build the next DCell level/layer. Thanks to its design, this topology leads to considerable scale-up. Also, `DCell` topology offers a high number of paths which enhancing the bandwidth, unsure the fault tolerance and improve routing particularly for multicast flows. Nevertheless, `DCell` requires a high number of network interfaces and ports when scaling which can have an impact on its cost. Also, servers in the lower layers deal more with transmitting packets which can engender load balancing issues.

`FiConn` [42] is a modified `DCell` topology proposed in 2009. The suggested design is similar to the `DCell` by deploying a compound graph to create its structure. This design based on the connection between servers and other devices via idle ports in opposite to the full connection used in `DCell` topology. Hence, the decrease of this servers network adapters engender a reduced number of required ports for the higher level switches. Consequently, the deployment cost of this architecture is minimized.

Simliarly to `DCell`, `BCube` [41] topology has a recursive defined architecture. Typically, the latter can be formed by n servers connected to n-port switches forming the $BCube_0$ units. Then, $BCube_1$ is composed by $nBCube_0$ attached to n switches forming the second level of the design as shown in Figure 2.8 `BCube` topology aims to deploy Modular Data Center (MDC) based

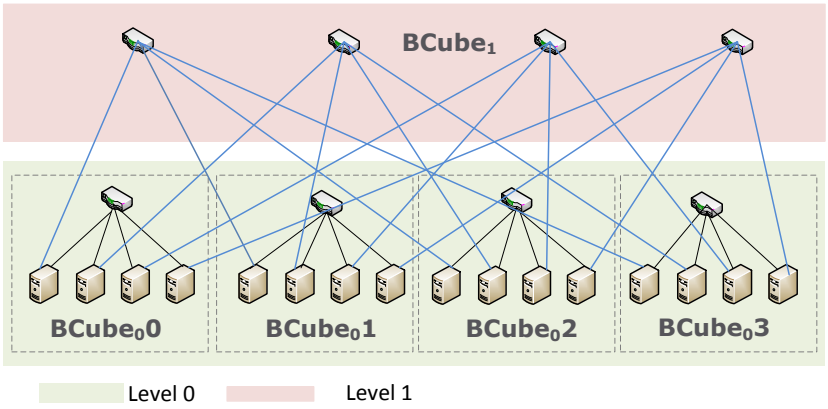

Figure 2.8 – Two-Levels Bcube DCN Topology

on shipping-containers data center in order to ensure high scalability. Hence, MDCube [81] data center was proposed to establish the interconnection between these data centers via fiber cables and to form a larger sized data centers. Thus, to form this HyperCube DCN, the BCube container is virtualized and interconnected with other similar virtual nodes. BCube servers present one of the most important asset of the BCube design. In fact, these multi-port devices can afford a high fault-tolerance, perform adaptive-routing and provide an important throughput and considerable load balancing especially in lower level servers. However, despite of its benefits, BCube topology still suffers from expensive cabling complexity and construction in contrast with DCell.

To summarize, compared to tree-like DCNs, Hybrid topologies decrease the number of layers and the cabling complexity. In addition, Hybrid topologies can offer the same bisection bandwidth[3] for same number of servers with a minimized number of switches and power consumption. However, these recursive designs still suffer from the single point failure, intensify the control plane complexity and present an expensive infrastructure by deploying high-radix switches.

2.2.3 Enhanced DCNs designs: Overview

The above mentioned wired architectures still suffer from flexibility and congestion issues despite of the several proposals designs based on high performed wired cabling. Therefore, researchers proposed a novel network scheme aiming to enhance the performance of these in-

[3]The bisection bandwidth is equal to the minimum of bandwidth required of all links capacities within a bisection. The latter is considered as a partition of an equally-sized number of nodes within the network [55]

frastructures. Hereafter, we will detail some relevant promising investigated solutions based on optical and wireless connections, and randomly connected devices within DCN.

2.2.3.1 Optical DCNs

We distinguish two types of designs in this category: Full Optical and Hybrid Optical.

In the first category, full Optical-DCN architecture i.e., O-DCN deploys optical devices in both data and control planes equipment in order to enhance the traffic speed and to boost the bandwidth within DCN. We note the Optical Circuit Switching i.e., `OCS` [82, 83] is characterized by performed core switches offering wide bandwidth. These data center relies on pre-configured routing path in the switch to ensure the maximum of required bandwidth [84]. Besides, for the second category, we cite the most relevant designs of the Hybrid Optical topologies which are `OSA`, `Helios` and `c-Through`. So, the authors in [85] addressed the Optical Switching architecture i.e., `OSA`. The latter is based on a central OSA manager that handles traffic management, infrastructure administration, and routing assessment and configuration. The `OSA` topology is based on the shortest path routing and optical hop-by-hop switching to establish the network connectivity. Also, `Helios` [86] and `c-Through` [87] make use of optical and electrical devices i.e., switches to ensure the traffic routing within the DCNs as depicted in Figure. 2.9. In fact, in these hybrid optical topologies, each ToR communicates with both electrical and optical networks meantime. The electrical network is defined by 2/3 hierarchically layered tree infrastructure with a defined over-subscription ratio, whereas, in the optical network, each ToR communicates through a single unlimited-capacity optical link with other ToRs. Despite of the enhanced performance offered by `Helios` and `c-Through` topologies, they still suffer from several limitation related to the routing overhead.

To conclude, the Optical switching infrastructure defines the ability of ensuring the high-speed traffic within a data center. In fact, thanks to the installation of optical devices/ links, these DCNs are able to satisfy Cloud services requirements by affording a highly flexible bandwidth with a reduced power consumption. Nevertheless, to avoid switches overhead, O-DCN architecture requires a challenging work consisting on deploying several modulation techniques in order to accurately adapt bandwidth while switching communication. Also, these designs can engender an important reconfiguration latency which can impact the performance of deployed cloud services within it. Finally, O-DCN topologies are so expensive for a large scale designs because of their high costly deployed devices such as optical transceivers.

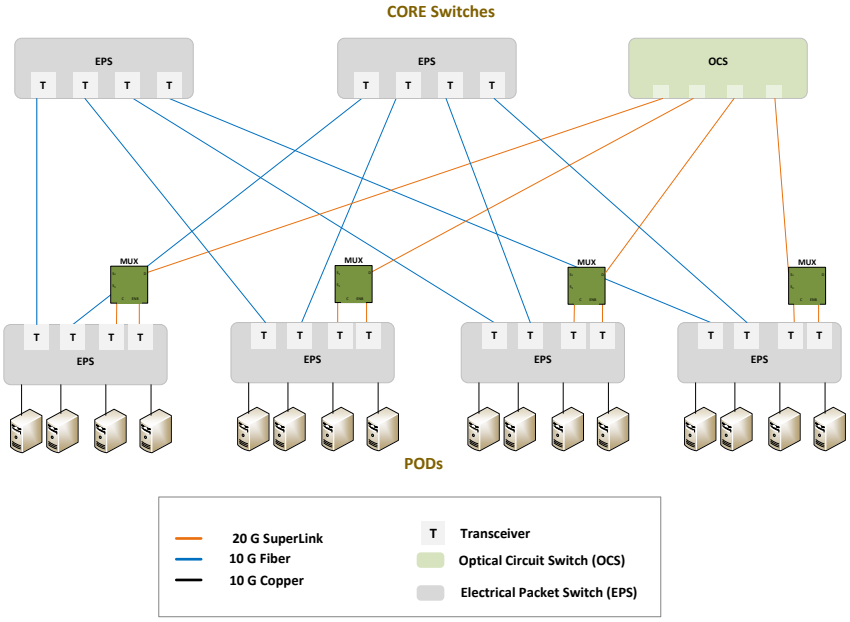

Figure 2.9 – 2-level of multi-rooted Helios tree topology

2.2.3.2 Wireless DCNs

To overcome cabling complexity and to alleviate the congestion issues, the wireless based-data center have been proposed. In fact, researchers suggest to extend the existing wired links suffering from bottleneck by wireless devices operating in the range of 60 GHz. Similarly to O-DCN topologies, the wireless DCN are classified into two basic categories: i) full wireless DCNs and ii) Hybrid wireless/wired DCNs. Hereafter, we overview some of novel propounded data center architectures belonging to each of the aforementioned categories.

For wireless/wired DCNs, we cite `Flyway`[88] [89]data center illustrated in Figure. 2.10. This architecture was proposed to minimize the data forwarding delay between ToRs switches. Thus, to mitigate the congestion for VL2 architecture. `Flyway` architecture deploys on demand links when a congestion occurred at the ToR level. Nevertheless, the proposed scheme failed to find a trade-off between all data center requirements such as sclalability, high traffic load and fault tolerance. Later, `WDCN` was proposed in [90] in which each ToR is supplied by 60 GHz wireless devices and is defined as Wireless Transmission Unit (WTU). The proposed architecture aims to manage the unbalanced traffic and to alleviate the node congestion. Moreover, `Angora` [91] wireless/wired data center was proposed. The latter design aims to transmit data through wired connection whereas the control plane is handled by a powerful wireless infrastructure. In fact, `Angora` design makes use of a 3D beamforming radios [92] built through Kautz graph [93] and

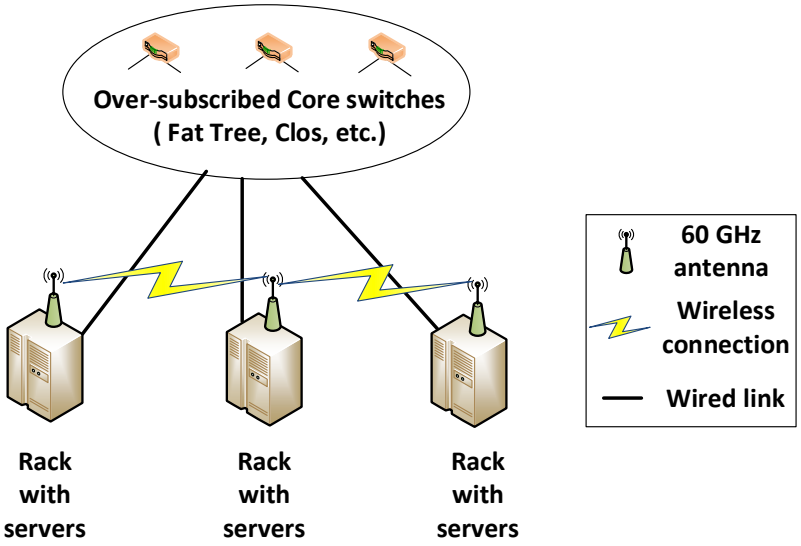

Figure 2.10 – Sample of Flyway augmented network topology

implemented on racks. This technique ensures a robust decoupled path from the network and a flexibility to establish the workload with low-cost wireless devices. However, despite of the low-latency control path provided by `Angora`, this topology engenders a high radio and racks failures. Besides, the authors in [94] proposed the `RUSH` data center which consists on the typical hierarchical three-layered tree infrastructure. The new design scheme is based on deploying a single directional 60 GHz antennas in each ToR. Also, a wireless connection was established by `RUSH` topology to interconnect racks in order to alleviate the congestion.

For full wireless data centers, we cite `Cayley DC` [95] where servers are arranged through cylindrical racks following the Cayley graph [96]. `Cayley DC` consists on deploying wireless connections for both intra and inter-racks communications following a mesh architecture. In fact, the proposed design aims to maximize the number of active links and to minimize the interference by deploying beamforming approach based on permanent-direction antennas.

In conclusion, implementing wireless connection allows flexible DCNs and minimizes cabling layout complexity. Nevertheless, for a defined bandwidth, these wireless designs can provide limited forwarding distance and can engender overhead due to the broadcasting. Also, for a high scaled wireless DCNs, these infrastructure can suffer from interference issues which can impact the performance of running services within it.

2.2.3.3 Randomly Connected DCNs

These Data center architectures are Considered as unstructured DCNs where servers and switches can be randomly connected to other device i.e., switch. Hereafter, we detail two main topologies in this class: i) `Jellyfish` and ii) `Scafida`.

`Jellyfish` [53], shown in Figure. 2.11, is an improvement of Fat tree topology. At the ToR switch level, the propounded design constructs a random graph to reach a flexible and dynamic network size. In fact, `Jellyfish` selects randomly a non-adjacent ToR switches with idle ports and attach them via a link. This operation will be repeated until no possible new link can be added. In contrast to Fat tree topology, `Jellyfish` can maintain full bandwidth 27% servers with the same switching devices for a maximum of 900 servers sized DCN [97]. Also, the latter design presents a shorter path length for routing than Fat Tree data center. However, this design still suffers for almost the same weaknesses as its predecessor such as switch failure recovery and limited network scalability due to a reduced number of core routers.

The authors got inspired by scale-free networks and propounded `Scafida` in [98]. This design is based on Barabàsi and Albert algorithm [99] to generate a scale-free network and to minimize the path length for routing in DCNs compared to other same sized architecture [100].

Summarily, the main advantage of the randomly constructed DCNs is providing low-latency and important bandwidth which can improve the performance of these infrastructures. However, these designs present a high cabling complexity and routing for a physically large scaled deployment.

2.2.4 Comparison between Data Center Fabrics

In this section, we propose a qualitative comparison between different DCN architectures. Table 2.1 summarizes the several DCN infrastructures according to the most important features for their designs, as follow:

- **Category**: we detail the class of each data center design following our proposed Taxonomy depicted in Figure 2.1. Also, we define each class in the table as follows:
  - Switch Only Data center as *Sw.O*.
  - Server Only Data center as *Serv.O*.
  - Hybrid/Recursive Data center as *Hybrid/Recursive*.
  - Full Optical Data center as *Full-O*.
  - Hybrid Optical Data center as *Hybrid-O*.

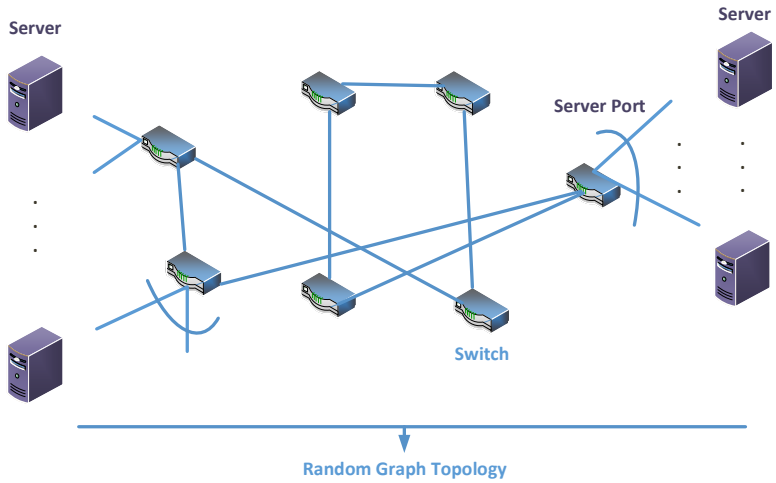

Figure 2.11 – Sample of Jellyfish topology

  - Hybrid Wireless Data center as *Hybrid-W*.
  - Full Wireless Data center as *Full-W*.
  - Random Data center as *Random*.

- **Connectivity technique**: defines the way how data center devices are connected.
- **Scalability:** It is considered as the most important criteria for designing DCNs. In fact, current architecture must provide the ability of interconnecting thousand of cloud computing services. Also, it must allow an extended network able to host the equipment upgrade without requiring a substantial re-organization within the topology.
- **Agility:** defines the ability of the data center to rapidly find an operable solution in order to ensure a fast failure recovery or to avoid DCN services interruption.
- **Resiliency:** defines the elasticity and the flexibility to maintain and enhance services of a data center despite of unexpected failure or environment changes e.g., network expansion [101].
- **Energy Consumption:** defines the overall energy consumed by DCN devices. This feature is important when conceiving this building in order to evaluate their impact on the ecosystem and the cost.
- **Cost of deployment:** qualifies the cost of installed devices such as switches, servers and cables used to design the DCN.

- **Bandwidth utilization:** Current Cloud applications define the appropriate bandwidth for each user. Therefore, this feature is highly required for DCN conception in order to outfit the users need and to respect the Cloud SLAs.
- **Fault tolerance:** illustrates the ability of a data center services to continue performing their functions even when a failure happened by avoiding any eventual system downtime [102].
- **Cabling complexity:** qualifies the multitude of deployed links when designing the DCN.
- **Load balancing:** This feature is important for defining the performance of the DCN. As an example, cloud architects should consider this criteria to avoid server bottleneck which can have a bad impact on the performance of cloud applications by increasing latency and packet loss.

Table 2.1 – Qualitative Comparison and Summary of Data centers Architectures

Ref.	DCN Name	Category	Connectivity Technique	Scalability	Agility	Resiliency	Energy Consumption	Cost of Deployment	Bandwidth Utilization	Fault Tolerance	Cabling Complexity	Load Balancing
[49]	CLOS	Sw.O	Ethernet wired	medium	yes	medium	high	high	medium	bad	medium	bad
[58]	Fat Tree	Sw.O	Ethernet wired	medium	yes	medium	high	medium	quite high	bad	medium	bad
[27]	Portland	Sw.O	Ethernet wired	high	yes	medium	medium	high	high	good	medium	quite high
[72]	Elastic Tree	Sw.O	Ethernet wired	medium	yes	high	low	medium	high	medium	high	high
[73]	Monsoon	Sw.O	Ethernet wired	medium	yes	medium	medium	high	high	medium	very high	high
[28]	VL2	Sw.O	Ethernet wired	medium	yes	medium	medium	high	quite high	medium	very high	quite high
[75]	FBFLY	Sw.O	Ethernet wired	medium	yes	high	low	medium	quite high	good	low	high
[76]	C-FBFLY	Sw.O	Optical wired	low	yes	high	medium	medium	high	good	low	high
[44]	CamCube	Serv.O	Ethernet wired	high	no	very high	very high	medium	very high	good	very high	very high
[80]	SRV deBruijn	Serv.O	Ethernet wired	high	yes	very high	very high	low	very high	good	high	very high
[40]	DCell	Hybrid / Recursive	Ethernet wired	high	no	high	very high	high	high	good	very high	quite high
[42]	Ficonn	Hybrid / Recursive	Ethernet wired	high	yes	high	very high	medium	high	good	very high	medium
[41]	BCube	Hybrid / Recursive	Ethernet wired	very high	yes	high	very high	high	very high	good	very high	very high
[81]	MDCube	Hybrid / Recursive	Ethernet wired	very high	yes	high	very high	very high	very high	good	very high	high
[82]	OCS	Full-O	Optical wired	medium	yes	medium	medium	very high	high	medium	high	medium
[85]	OSA	Hybrid-O	Optical/Ethernet	low	no	bad	medium	medium	high	bad	high	bad
[86]	Helios	Hybrid-O	Optical/Ethernet	medium	no	bad	medium	high	high	bad	high	bad
[87]	C-Through	Hybrid-O	Optical/Ethernet	medium	no	bad	medium	high	high	bad	high	bad
[89]	Flyway	Hybrid-W	60 GHz/Ethernet	medium	yes	medium	medium	high	medium	medium	medium	quite high
[91]	Angora	Hybrid-W	60 GHz/Ethernet	medium	yes	high	medium	medium	high	good	medium	quite high
[94]	Rush	Hybrid-W	60 GHz/Ethernet	medium	yes	medium	medium	high	high	medium	medium	quite high
[95]	Cayley-DC	Full-W	60 GHz antennas	medium	yes	medium	medium	very high	medium	medium	high	medium
[53]	Jellyfish	Random	Ethernet wired	high	yes	high	very high	high	very high	good	very high	quite high
[98]	Scafida	Random	Ethernet wired	high	yes	medium	very high	high	very high	good	very high	quite high

2.3 Conclusion

In this chapter, we overviewed the existing data center architecture proposed in the literature. Then, we classified these DCNs into major categories depending on the design and the role of network devices within it : i) typical Data Center Architectures and ii) Enhanced Data Center Networks. Afterwards, we summarized the deeply analyzed architectures and we proposed a qualitative comparison according to the most important features typically considered for DCNs designs. In the next section, we will detail the state of art of routing protocols for each class of the aforementioned DCN architectures.

Chapter 3
Taxonomy of Routing in Data Center Networks

Contents

3.1 Introduction

It is undeniable that researchers are proposing different data center architectures aiming to not only enhancing network QoS metrics but also improving the routing of exponentially growing intra-data center traffic. But, the latter suffers from the complexity of deployment and requires high network performance to fulfill customers QoS need. However, the customization of such a network still possible since it is under one cloud operator [23]. In this chapter, we summarize most the relevant and recent proposals, existing in the literature, addressing the routing problem for both categories of data center topologies: i) Traditional DCNs and ii) Enhanced-DCNs designs.

3.2 Unicast Routing within DCNs topologies

Data center networks operations aim to transport data between source and destination nodes through several defined criteria such as load balancing, bandwidth availability, efficiency of energy consumption, etc. Switch-only based data center topologies [23] (e.g., Clos, Fat-Tree, Portland) need efficient routing protocols to exploit the multiple alternative paths and to enhance the QoS. In fact, switches are the only relay nodes in these fabrics. In this section, we investigate proposals implemented in order to fit the aforementioned objectives for one-to-one traffic communication i.e., Unicast protocols within DCNs.

3.2.1 Unicast Routing for Typical DCNs

Switch-only data center suffers from loops when broadcasting packets. In fact, these fabrics rely on switches to relay packet. However, to handle loops issue means that we have to eliminate broadcast in the network. Consequently, dealing with automated path selection through switches presents a critical problem. To overcome this issue, some proposals rely on the Spanning Tree Protocol (STP) [103] which eliminates the redundant non-used paths and generate a spanning tree for the packets routing within network. Thus, STP provides a bisection bandwidth[1] improvement. Similarly, the authors in [104] addressed the packet loops problem for switch centric DCNs e.g., Clos and Fat tree. In fact, Fat Tree based data centers can use Equal Cost MultiPath (ECMP) protocol [104] for routing packet between switches. Therefore, ECMP can provide a better utilization of alternative paths than single path routing offered by the two-phased algorithm in Clos topology and alleviate the packet loops problem within switches. In the same context, the authors in [105] propound a novel routing strategy based on the Ternary Content Addressable Memory (TCAM) technique. The latter technique consists on adding a second TCAM Switch table within switches in order to interconnect severs belonging to the same TOR switch to other servers in specific pods. The proposed model showed its efficiency in terms of security and QoS. Thus, the authors made several experimentations to test the performance of their proposal by comparing it to the existing techniques in the literature. Besides,

[1] The bisection bandwidth is equal to the minimum of bandwidth required of all links capacities within a bisection. The latter is considered as a partition of an equally-sized number of nodes within the network [55]

the authors confirmed that the proposed TCAM Switch technique can help to solve TOR and Leaf switches limitations such as providing an important throughput, avoiding link oversubscription and Packets loop in addition to enhancing TCP incast and outcast for a high scaled data centers. Through some validation tests, the authors confirm that TCAM provides a negligible on-ship power consumption (less than 3%) and performance overhead. Also, similarly to the Simple Switch, TCAM recorded a same line-rate throughput and latency. But, despite the aforementioned prominent results, TCAM needs to be tested in real expended multi-tiered DCNs in order to evaluate the performance and its cost. In fact, multi-tiered and Clos topologies can have an expensive cost when scaling due to the need of deploying a high-cost spine switch that have very high capacity ports [106]. Moreover switch-centric data center topologies are not efficient in large-scale for several traffic patterns such as one-to-all or all-to-all-flows. Hence, these fabrics still suffer from bandwidth bottleneck, single failure switch and network congestion e.g., for routing long flows with ECMP [107].
Therefore, hybrid topologies such as DCell, Bcube and BCDC can overcome these issues and provide better bandwidth availability and augment the fault-tolerance between servers paths in the context of switch failure [108].

Indeed, for hybrid topologies, the authors in [109] proposed the LaScaDa Topology which is considered as an enhancement of BCube and HyperCube topologies. Besides, the authors addressed two novel routing strategies schemes in LaScaDa DCNs for routing unicast traffic namely Fault-Free Routing Scheme and Fault-Tolerant Routing Scheme. Through several experimentations, the authors showed that the proposed topology outperforms the existing switch and hybrid DCNs such as DCell, BCube and Fat tree in terms of QoS performance i.e., bisection bandwidth and scalability. In the same way, authors in [108] show that the proposed BCDC DCN can provide the aforementioned performance and even outperforms DCell and BCube topologies in terms of fault tolerance while simplifying the DCN deployment. Hence, considering these requirements, the authors propound a new fault tolerant routing algorithm BRouting based on the shortest path within BCDC. By the end, the authors demonstrated that the proposed routing algorithms can deeply impact the unicast, multicast and broadcast communication rapidity. Through several experiments, the authors proved that BFRouting algorithm can provide a good performance by carrying out distributed fault tolerant paths while avoiding the use of global state. Also, simulations showed that BCDC performs a high network extension. Also, the proposal can afford a good performance and controls the deployment capacity respectively in terms of fault tolerance and cabling complexity. In the same context, the authors in [110] made use of BFRouting to design their proposal in order to address the fault tolerance routing and node-disjoint paths for BCDC topologies. The suggested proposal tackled the unicast, multicast and broadcast routing communications. For the unicast traffic, the authors proposed BRFT Routing algorithm running under 1-restricted connectivity based on the fault-free paths and relying in Breadth-First Search BFS. For performance analysis, the authors performed a node disjoint-path and fault tolerance 3-dimensional BCDC DCN. Results show that the algorithm can not exceed $3n-5$ of faulty nodes where each one can have at least a fault-free neighbor in the DCNs. Consequently, the proposed BCDC offered a better traffic routing than DCell and Fat Tree DCNs in terms of node-disjoint paths and fault tolerance.

Server-only data center topology such as the torus is characterized by its large path diversity which engenders a high number of equal cost paths. In the proposed architecture (mesh,

torus, hypercube, CamCube, etc.), typified routing algorithm are proposed by considering the specificity of network topology. To the best of our knowledge, there is no recent study tackling the routing within CamCube and/or server only topology. Therefore, we cite hereafter the existing studies published respectively in 2010, 2013 and 2014. In [44], the authors proposed the symbiotic routing protocol based on multi-hop routing service using link-state shortest path algorithm. The communication between servers, to forward packets, is based on the call-back function to inform the running service about the source link on which the packet was received. The routing process is running in servers as services where applications adopt their own routing strategy for resource and performance optimization. In [111] [112], the authors proposed a key-based routing protocol where keys determine the coordination space which designates the location of the servers within it. However, the transmitted packet is delivered only when the destination server is reachable. Otherwise, the packet is dropped [112]. This can lead to increase the packet loss rate especially without re-transmission mechanisms in a large scale DCN.

3.2.2 Unicast Routing for Enhanced DCNs

Different data center designs have been proposed to investigate the performance of data routing in the Optical DCNs. In [113], the authors addressed the wavelength routing within POI architecture by analyzing the DCN scalability and its crosstalk specifications. The authors performed different traffic patterns and made a cross-layer study based on k-shortest path algorithm within Passive Optical Interconnect (POI) architecture. Also, this fabric is built by symmetric and non-blocking wavelength routing devices. According to the authors, POI architecture can have theoretically a similar behavior as DCell topology while specifying proper port configuration. According to their study, the authors showed that the network performance, in a such topology, is impacted by the physical layer implementation i.e., optical interface configuration and transceivers design. Also, authors deduced that bidirectional optical interface can be more appropriate in order to increase the performance of the highly extended network. Finally, to provide a trade-off between transceiver design complexity and network performance, the authors suggest the replacement of array-fixed receivers by few tunable ones. In fact, results show that by reducing the half number of receivers, we can reach a small augmentation of blocking probability.
Similarly, the authors in [114] studied the improvement of blocking probability and the spectrum consumption for multiple path routing within Elastic Optical DCN (EO-DCNs). To do so, the authors propound the Multipath Provisioning with Content Connectivity (`MPCC`) strategy. For the static traffic context, the authors proposed the ILP formulation to solve `MPCC`, whereas, they made use of heuristic algorithms to solve the proposed scheme within dynamic traffic scenario. Through several simulations, the proposed `MPCC` achieved the study goal by decreasing over 20% of spectrum consumption in comparison with the existing multipath provisioning strategy for both static and dynamic traffic environment. Besides, the blocking probability is reduced to the half (50%) for the dynamic traffic strategy. Finally, the authors deduced that `MPCC` is resilient for the users requests variation when it relies on the Dynamic Content Placement (DCP) strategy.

Furthermore, considering routing within Hybrid wireless DCNs, in [115] the authors tackled the Proactive Retention aware caching through multipath Routing within hybrid wireless edge

networks in which mobile users access to servers via one or more edges caches. Then, the authors proposed a non-convex and non-Linear Mixed-Integer Programming (non-MILP) model to formulate the Proactive Retention Routing Optimization PRRO problem. Besides, the authors address a heuristic solution to tackle the cache capacity case. In fact, by their proposal, the authors aimed to minimize the cost of content storage cache and server access. Finally, through systematic performance evaluation, the authors deduce that the proposed algorithms provides near-to-optimal solution in comparison with the existing caching strategy for both unicast and multicast traffic routing.
Similar to the latter proposal, authors in [116] addressed the unicast and multicast routing modes within DCNs by proposing the XOR-based Source Routing (XSR) scheme. The authors aimed to perform rapid packet forwarding and facilitating its deployment while decreasing the latency communication. Indeed, XSR aims to enable a rapid decision computation and to establish the path labels for both treated modes. Thus, the objective is to efficiently improve routing packets within the network while keeping the routed packets unchanged for forwarding. The proposal was compared to the existing standard table lookup and recent modular arithmetic strategies for the performance evaluation. Moreover, for unicast mode, the proposed XSR allows to the receiver to use the same path label established from the source. Finally, obtained results show that for both unicast and multicast routing, the XSR offers the smallest label for computation and is considered as a highly scalable source routing scheme able to be implemented in any environment i.e., independent from the DCN infrastructure and the deployed core vendor within it. But, regardless the prominent obtained results, the proposed XSR should be implemented in a real environment e.g., by using Mininet emulation for tests in order to analyze the cost effectiveness and discuss the efficiency of the routing scheme in a real world simulations.

Concerning the Multicast traffic routing within DCN architectures, we investigate in the next section the recent proposals addressing this issue in the literature.

3.3 Multicast Routing within DCNs topologies

Multicast routing becomes more useful to save network traffic and avoid repeated transmission tasks for the sender [65]. Consequently, it is considered as a hot topic and several research works have been published in this field.

3.3.1 Multicast Routing for Typical DCNs

Traditionally, switch only and hybrid data center network topologies deploy switches and/or routers to forward packets between active servers. But, this kind of structures suffer from several problems to route efficiently packets. In fact, these relay nodes are bandwidth-hungry and limited routing space with narrow forwarding/routing tables (e.g., less than 1500 entries)[117].

To overcome this issue, in [118], the authors proposed the Intelligent Rendezvous Point (iRP) algorithm to tackle the efficiency of the construction of multicast trees while maximizing the residual bandwidth in Clos links. The authors made use of SDN-based system to implement the proposal and compared it to the existing control protocol i.e., Protocol Independent Multicast

PIM[2] in Clos fabric. Through several simulations, the authors deduced that the proposed iRP is better than PAM in term of bandwidth utilization. Thus, multicast flows are accepted by the system according to the availability of the bandwidth while avoiding the over-subscription in some Clos links. Also, obtained results confirmed that iRP system can enhance the multicast flows management within the DCNs and provides up to 50% of the Clos capacity enhancement. Finally, the authors confirmed that the proposed iRP achieved the security requirements by offering the possibility of accepting or denying the arrived flows in the system. In the same context, the authors in [119] tackled the efficiency and the capacity of multicast trees in Clos topology by proposing Learning based Intelligent Rendezvous Point (LiRP) in addition to (iRP) algorithm. The authors addressed the offline and Online optimization of the multicast tree. In fact, the authors assumed the ILP model in order to fulfill the required bandwidth in Clos link while optimizing the distribution of the trees in the network and to enhance the overall DCN utilization. Both algorithms aimed to improve the multicast tree construction within the DCN while ensuring a good load-balancing of the arrived multicast flows. Also, iRP and LiRP provide multicast flows admission control depending on the availability of the links capacities in terms of bandwidth. LiRP makes use of neural network for optimizing the multicast routing by predicting the multicast groups. Consequently, results show that LiRP outperforms iRP for enhancing the efficiency of the DCN. Regarding the use of online flows, results indicated that LiRP can provide a close performance to the optimal offline optimization. However, the proposed testbed needs to perform different online and offline optimization techniques in order to make a full comparison and consequently improve the efficiency of the proposals. Furthermore, the authors in [120] propose the Distributed Intelligent Rendezvous Point DiRP for the same aforementioned environment in order to treat the Multicast Tree construction Problem (MRP) [121]. The addressed study aims to increase the multicast capacity in the DCN and to overcome the limitations presented by PIM such as the absence of bandwidth awareness. The obtained results indicate that DiRP achieves the authors goal by enhancing the performance of the system up to 60% higher than PIM with a stable path setup delay.
Moreover, for the Leaf Spine networks, the authors in [122] aimed to build optimized efficient multicast trees. To do so, the authors proposed two algorithms to generate and recover the multicast tree relying on the `greedy set cover` algorithm [123] in order to find the near-optimal spine switch within the fabric. By the proposed algorithms, the authors confirm that results can significantly decrease the traffic load on uplinks. Finally, the authors considered that for this context, the optimization of multicast tree can be the same as the minimum set cover problem. Besides, considering the Fat tree DCNs, the authors in [124] addressed the multicast traffic within DCNs where multicast traffic sources presented by IoT devices. In this context, the authors tackled the problem of balanced traffic distribution when deploying different IoT devices in the network. To do so, the authors proposed to develop a new optimization algorithm allows balanced throughput and traffic delay for a dynamic big data broadcast. Consequently, the authors designed a new Dynamic Big-data Broadcasting approach (D2B) based on single-leader-multiple-follower Stackelberg game in which switches play the role of leader and IoT devices the followers. For each multicast traffic, the source node is broadcasting data in real time. For the analysis study, the authors evaluate theoretically the existence of the generalized

[2] PIM is a set of protocols that provide several delivery mechanisms such us one/many-to-many communications.

Nash-Stackelberg equilibrium for the proposal. Besides, in order to evaluate the performance of the proposal, the authors compared D2B scheme with two existing strategies for DCNs i.e., LSBT [125] based big-data broadcasting and the fat tree multicast DCNs [126]. Obtained results show that the proposed scheme outperforms the two compared strategies by increasing the throughput up to 55.32% and the bandwidth allocation to 33% while minimizing the overall delay for the studied DCN. Similarly, in [127] the authors showed the importance of designing an optimized-layer2 protocol in order to improve the reliability and the traffic separation within DCNs by generating disjointed multiple routing paths for multicast frames. To do so, the authors designed the Discovery of Multiple Disjoint Paths scheme (DMDP) that is able to select the disjointed paths while avoiding routes calculation within the network graph. Consequently, in order to evaluate the proposal, the authors compare DMDP using Mininet SDN platform to a second installed environment based on Naive Link-disjoint version of Bhandari's algorithm (Naive LBA) [128]. The obtained results show that DMDP outperforms the Naive LBA by providing between 60% to 97% of effectiveness ratio. Also, DMDP generates multiple paths similar to k-shortest path algorithm with a simplest way to find the resulting routing path. Finally, the authors deduced that DMDP offers simple mechanisms for detecting the multiple path routing while eliminating the path computation and ensuring low complexity for deployment.

According to the aforementioned studies, the authors tackled the multicast traffic issues basically in switch-only and/or hybrid topologies. By these proposals, authors aimed to balance the bandwidth utilization, increase the throughput and decrease the delay and the packet loss within network. However, to the best of our knowledge, we have not found in the literature study addressing these issues in server only DCNs i.e., CamCube Server.

3.3.2 Multicast Routing for Enhanced DCNs

Some research studies addressed the multicast routing for Enhanced Data Center Networks. Hereafter, we summarize the most relevant proposed solutions for multicast routing found in the literature.

For hybrid wireless data centers, in [129] the authors aimed to ensure both high performed multicast routing and wireless gains within Hybrid DCN (HDCN). To do so, the authors propound two Novel Efficient Multicast Routing schemes named respectively NEMO-Group and NEMO-cluster. The objective of the proposals is to provide a rapid construction of the multicast tree while simplifying the network traffic routing. The authors deploy the presented proposal within a wireless/Fat tree topology to analyze the performance of the suggested multicast routing scheme. The proposed topology is characterized by a high-throughput typical wired network and 2D/3D in 60 GHz beamforming wireless links. By NEMO-Group and NEMO-cluster, the authors aimed to build a weight-aware minimum spanning multicast tree that depends on the ratio α. The latter represents the ratio of wireless weight to wired weight. The authors considered that by minimizing α the routing speed increases. Also, the authors select the shortest path between source and destinations in order to minimize the number of intermediate nodes. Consequently, the cost and congestion of the traffic for multicast routing will be controlled and reduced. For the performance evaluation, the authors made qualitative comparisons of the proposals to justify the efficiency of the resulted multicast trees. After several simulations, the proposed algorithms reached the work objective by saving the multicast traffic within hybrid

wireless data centers i.e. by $15-40\%$ and enhancing the network throughput up to $10-40\%$ in comparison with the existant solution deployed in traditional wired DCN, i.e., Efficient and Scalable Multicast routing (ESM) [130].

For the Elastic Optical data center networks (EO-DCNs), the authors in [131] addressed the multicast routing for geographically distributed data centers and proposed a distributed sub-tree based optical multicasting scheme DST-OM. The latter is formulated as an integer linear program that aims to reduce the total spectrum consumption of all multicast demands in the EO-DCNs. Moreover, the authors proposed the Minimum Spectrum Sub-Tree problem MSST and implemented heuristic algorithms to address it. After several numerical comparisons with the conventional common source sub-tree and the single-tree based optical multicasting schemes, DST-OM scheme shows higher efficient spectrum and lower non-blocking probability. Several other studies tackled the multicast routing within EO-DCNs. In fact, the authors in [132] tackle the addressed problem through three directions. The first aspect consists on considering the most spectrum-efficient protection i.e., path-based protection in EO-DCNs. In a such environment, the authors confirmed the availability of multicast service access from several locally-distributed DCNs. Second, the authors made use of the distance-adaptive spectrum allocation in order to solve the Routing, Modulation Level, and Spectrum Assignment (RMLSA) [133] problem. Finally, the author addressed the Protection Distributed Sub-Light-Tree based on Shortest Path (PDSLT-SP) and evaluated its performance through ILP and efficient heuristic approaches in the context of static and dynamic traffic scenario within respectively small and large scaled EO-DCN. Thanks to the proposed PDSLT-SP, results of several simulations showed that the former proposal can provide an augmented modulation format while minimizing the spectrum consumption. Also, in order to overcome the link failure, results showed that the addressed proposal can provide an alternative link-disjoint path to protect each source-destination communication. In conclusion, the evaluation of performance confirmed that PDSLT-SP outperforms the conventional schemes in terms of blocking probability and spectrum resource utilization by offering a reasonable cost while using transmitters. In the same context, the authors in [134] addressed the Multicast Routing and Spectrum Assignment (MRSA) problem for static and dynamic scenario within EO-DCNs. To solve the studied problem, the authors formulated a joint and separate ILP models and suggest three heuristics algorithms respectively designed as CMRSA, DMRSA and mixed CMRSA/DMRSA. The latters heuristics aimed to avoid network congestion when performing the selection of multicast paths. The performance evaluation was studied through different type of scenario: i) static and ii) dynamic. For the first one, results show that joint ILP outperforms the separate ILP and the suggested heuristics in terms of spectrum assignment efficiency. However, the joint ILP presents the worse results in terms of run time. Also, the mixed CMRSA/DMRSA show better results than separate ILP in terms of weighting coefficient. Considering the second scenario, the experiments show that CMRSA improves the congestion while reducing the spectrum efficiency. Whereas, DMRSA improves the spectrum efficiency and reduces congestion while getting a high bandwidth efficiency. To conclude, the authors considered the mixed CMRSA/DMRSA as a better solution for making a trade-off between spectrum efficiency and congestion by minimizing the blocking probability. Through the numerous studies existing in the literature tackling the multicast routing problems in EO-DCNs, we can conclude that this issue still attracts the researchers' interests in order to improve the traffic management in data center designs particularly for modern enhanced fabrics.

Furthermore, for Optical DCNs, the authors in [135] tackled the transmission performance for the wavelength routing and optical multicast in data center networks. In fact, the addressed design aimed to allow a reuse of the spatial wavelength and scalable and modular multicast strategy in case of wavelength routing. To do so, the authors implemented a cross-layer framework in order to evaluate the performance of Pulse Amplitude Modulation (PAM) and code rate adaptation for wavelength routing and optical multicasting DCNs. According to their analysis, the authors enlightened the important impact of Arrayed-Waveguide Grating (AWG) crosstalk on higher order PAM within wavelength-reuse and distributed architecture. Through several simulations, the authors conclude that the 4-PAM provides a higher effective bit rates for interconnecting two broadcast domains and an important crosstalk stage. But, the authors deduced that 8-PAM provides a highest performance for the communications ending in one broadcast domain.

3.4 Batch Routing within DCNs topologies

To the best of our knowledge, the problem of batch routing of intra-data center flows specifically in wired typical fabrics has been rarely addressed by researchers. Also, we noticed that a few research studies addressed the batch problem scheduling for enhanced DCNs particularly hybrid (wired / wireless) DCNs. Hereafter, we summarize the main relevant works in this context.

3.4.1 Batch-Routing for Typical DCNs

Typically, the batch routed flows technique is used to optimize the treatment of a large number of arrived flows. According to the literature, the only existing solutions proposed by researchers are new routing schemes based on batched arrived flows only for Enhanced data centers. Consequently, to our best survey we can cite [136] and [137] for batched traffic in Typical DCNs. In [136], the authors propounded FCTcon as dynamic Flow Completion Time (FCT) for controlling the power optimization within DCNs. FCTcon is deployed to control the delay-sensitive traffic flows to fit the requirements and to guarantee a high performed networks by maximizing the power DCNs savings. The obtained results show that with 50K servers-DCNS, the proposal, achieved up to 62.2% extra net profits. Moreover, the authors proposed two designs as an extension of FCTcon to deal with the co-flow and batched flows by decreasing the co-flow deadline miss ratio between 12% and 15%. In [137], the authors proposed the Adaptive Request Schedule ARS as a novel cross-layer design deployed at the aggregation switches side for Clos topology. ARS provides a dynamic adjustment of concurrent TCP flows for batched application requests. The proposed ARS is broad applicable and was deployed without modification with Data Center TCP (DCTCP) and TCP New Reno on both NS2 simulator and small-scale Linux testbed. The obtained results confirmed that the proposed ARS has deeply minimized the incast[3] probability through several TCP protocols and has regularly augmented the network good put even for a

[3]Incast is many-to-one communication generally deployed in cloud DCNs for distributed storage or computing framework such as MapReduce

highly congested network. In [138], the authors tackle the problem of ECN[4] based on instantaneous queue length since ECN is highly deployed in the new generations of Data centers. The authors noticed that the threshold of ECN is low which can minimize the link utilisation capacities within DCNs and can cause switch buffer underflow. Besides, the analysis show that ECN mis-marking can cause an overreaction from senders. Thus, to overcome this problem, the authors propounded CEDM which is a Combined Enqueue and Dequeue Marking able to mark packets in a suitable manner. The resulted experiments showed that CEDM reaches the authors goal by minimizing throughput loss and enhancing the flow completion time.

Besides, the authors in [139] propose a new strategy to minimize the power consumption while satisfying the performance requirement for batch processing applications such as Spark for private or public cloud. For that purpose, the authors addressed the scheduler ExpREsS for orchestrating the execution of Spark applications and DVFS techniques to decrease the energy consumption. The authors made use of the Earliest Deadline First (EDF) algorithm to solve the addressed problem. The designed approach is workload agnostic and find out the execution order dynamically while ensuring the required resource for the requests of the Spark application.

In the next section, we will summarize the most significant studies tackling the batch routing within Enhanced DCNs e.g., hybrid wireless DCNs. Note that the existing proposals consist in improving the traditional data center infrastructure by adding wireless connection e.g., to connect the ToRs switches in Clos data centers via 60 GHz antennas.

3.4.2 Batch-Routing for Enhanced DCNs

Considering the hybrid (wired and wireless) DCNs, the authors in [140] proposed a new architecture based on Fat Tree DCN by adding a wireless infrastructure namely VLCcube. In this fabric, only inter-rack communication is established via wireless links by using the Visible Light Communication (VLC) techniques. The authors proposed a new routing scheme for each flow routed in a hybrid path for both online and batch mode. This proposal is specified to VLCcube topology where only routing issues have been considered. However, the authors unconsidered the interference constraints and channel allocation for optical wireless communications.
Moreover, in [141] the authors propounded a novel routing and congestion-Aware flow scheduling scheme for batched traffic with a newly implemented DCN called TIO. The latter is a hybrid DCN and addressed as a combination of wireless Visible Light Communication (VLC) based on Jellyfish DCN and wired Electrical Packet Switches (EPS) based on Fat Tree DCN using Clos topology properties. The proposed routing strategy aims to alleviate the network congestion and balance the traffic load through a scheduling method designed for the batched traffic pattern i.e., Batched Flow Scheduling BFS. Thus, the routing scheme proposes an optimized hybrid path for routing the traffic within TIO. The results showed that the use of a such flow scheduling deeply enhances the performance of tested networks i.e., in comparison with ECMP routing. Also, TIO solution outperforms Jellyfish and Fat tree DCNs in terms of topology properties and network performance.
Similarly, the authors in [142, 143] propounded a new framework that deals with online and batch routing schemes for HDCN topology based on Clos wired infrastructure. The path compu-

[4] ECN is an effective tool used in data centers where switches can notify senders in case of congestion.

tation is based on hop count and QoS metrics. Two solutions have been proposed: i) heuristic, denoted by JRBC-HDCN and ii) scalable denoted by SJB-HDCN in order to enhance throughput while reducing interference effects. So, the proposed approaches took into account the link state and then aimed to maximize the wired and wireless bandwidth interfaces in order to enhance the performance of the network. Unfortunately, SJB-HDCN is based on scalable Lagrangian relaxation solutions which are known for their high computation time so suffering from convergence time although providing a near to optimal solution.

However, the previous cited studies only address the wireless and optical resource allocation problem in enhanced DCNs which is not our focus since we tackle the resource allocation problem for only wired infrastructure of server-only topologies.

3.5 SDN Routing for Data Center Networks

Currently, one of the most important motivations of data center traffic engineering is to ensure a high reliability and QoS satisfaction by optimizing traffic forwarding. The physical data center infrastructure is administered and managed by only one operator. Consequently, it is easier to deploy a centralized solution based on the Software Defined Network (SDN) paradigm. SDN separates the control plane from the data plane. A central authority is able to collect in real-time the topology and the state of the network called SDN controller such as Open Networking Operating System (ONOS) [71] and OpenDaylight [144]. In addition, the centralized traffic engineering can be deeply improved compared with classical distributed related strategies in the view of the fact that the controller maintains a perfect view of the topology and the network state at a given time. Also, deploying a software centralized network control can enhance the network evolution and simplify the network management while adding new networking abstraction [145].
A full description of our proposed SDN-based solution is provided in Chapter 4. In this section, we will introduce the most significant strategies proposed in the literature for both typical and enhanced data centers using the SDN paradigm.

3.5.1 SDN Routing for Typical DCNs

Some research works addressed the software-defined control plane for managing the unicast and multicast traffic within DCNs. In this context, the authors in [146] proposed to analyse the TCP throughput and jitter for UDP traffic. The authors considered that deploying the SDN controller in a such topology can add more accessibility to the network while constructing the routing scheme. Moreover, the authors results are based on the comparison between the proposal i.e., ECMP based on a modified Dijkstra algorithm and the static routing within Fat tree topology. The authors made use of the modulo-n hashing method of the ECMP to deliver the resulted path. According to the authors, the proposal provides a better throughput and a minimized packet loss comparing to the existant routing algorithm within Fat tree topology.
In [145], the authors proposed the cRetor as a new SDN based topology aware routing approach for highly scaled DCNs. In a such design, the switches are equipped with a topology-aware pro-

cessors in order to forward packets efficiently in a fault-free topologies independently of the support of the SDN controller. The authors show that cRetor proposal can enhance the topology discovery and the path calculation within a regular topology. According to the authors, the aforementioned proposal can provide a high scalability, a fast failover and a high SDN manageability by assuring a compatibility with all SDN applications and a good QoS performance by reducing the packet loss. Note that authors considered regular topology where network devices are grouped according to their similar patterns, locations and connections. Also, the authors proposed a new routing scheme based on Topology Description Language (TPDL) and SDN topology. The authors compared the latter proposal with OSPF and conventional SDN network using floodlight and DCell. Results showed that the proposed routing algorithm provides a better convergence time and a fault recovery performance. Also, cRetor shows a better exploitation of the data center networks and increase the efficiency of the network. Besides, in [147], the authors propounded the Node Load Aware Dynamic Routing NLADR in order to improve the throughput and alleviate the congestion within the conventional DCNs. Moreover, based on the SDN concept and the Channel State Information (CSI) provided by the SDN controller, the NLADR protocol grasps the network real-time state of the DCN and allocates the resources accordingly.
In addition, in [148], the authors proposed the low-cost and load-balanced route management (L2RM) framework for Fat tree topologies. L2RM framework aims to verify the status of switches and links by monitoring network traffic in order to alleviate the heavy loads in the DCN infrastructure. Results of the Mininet simulations showed that the proposed framework provides a better link utilization and reduced table overflows and message cost in comparison with different SDN-based strategy already deployed for Fat tree DCNs. Furthermore, considering the multicast traffic flows for SDN-based DCNs, the authors in [149] design the Priority-based Adaptive Multicast PAM. The latter controls the sending rates of concurrent multicast transfers based on their priorities. They are defined according to some policies such as Smallest Remaining Size First (SRSF). By the end, results showed that PAM is near-optimal of priority-based schedules by decreasing the average transfer completions and outperforming the default fair sharing to up 5.7 times. Also, the authors deduced that PAM is low overhead and TCP-friendly. In fact, the proposed prototype has a negligible impact on TCP traffic and converges rapidly.
Finally, in [150], the authors propounded a novel routing scheme for SDN-based data centers that is based on deep Q-learning (DOQ) to provide an optimal routing paths based on intelligent decision for these DCNs. DOQ relies on the deep neural network to select the optimal path by rapidly retrieve the flow type and the input network state. The addressed problem consists on Markov Decision Process Problem (MDP) and authors propose Reinforcement Learning (RL), represented by Q-Learning(QL) [151], to solve it. The authors made use of mouse and elephant flows for the generation of the analysed traffic. The authors rely on the port rate and the flow table utilization for analysis. Results showed that the proposed routing strategy outperforms the existant ECMP routing and Selective Randomized Load Balancing (SRL) in terms of average delay and packet loss rate for mice-flows and average throughput for elephant-flows.

Generally, Typical DCNs rely on switches to forward packets. Unfortunately, regarding the success of SDN concept deployed in modern data centers in the latest years, switching device hardware needs to be more performed to outfit the increasing performance requested by today's SDN applications. In fact, SDN switching devices still require more memory space and better

processing speed to generate the tremendous traffic while being cost-effectiveness [152] [153]. Therefore, virtualized switches deployed in Server Only DCNs, or SDN design features with new integrated technologies deployed within Enhanced DCNs, could be a solution to overcome the aforementioned drawbacks.

3.5.2 SDN Routing for Enhanced DCNs

Integrating innovative technology in wireless and optical transmission can improve the coverage of SDN within DCNs [152]. In fact, by deploying such a new technology, the SDN controller can have a widespread among the whole network behavior and parameters. Hereafter, we cite some proposed routing scheme based on wireless and optical DCNs aiming to improve the SDN network management and QoS performance.

For SDN-based Hybrid Wireless Data center, the authors in [154] propose a both routing and forwarding scheme that aims to minimize the number of flow setup requests for these designs. In fact, to overcome the SDN scalability issues engendered by non-considering the flow characteristics, the authors suggest to compare their proposal with the existing approaches in both wired and wireless topologies. Results show that the proposed scheme outperforms the traditional approaches and characterized by interesting insights for the design of wireless DCNs based on SDN concept. Also, in [155], the authors propound a new greedy-heuristic scheme for wireless data centers in order to decrease the total update time. The authors start by proposing a wireless dependency model and formulate a resource competition problem by minimizing the update time. Then, the authors address a three-step greedy-heuristic strategy in order to provide an efficient and completed network update for a hybrid SDN wireless data center. The authors compare their proposal to the existing strategy and show that the addressed solution can provide a better performance by deeply reducing the update time and rule modification count.

Considering the Optical DCNs, in [156], the authors propose a novel cross layer of SDN control framework for the real-time tuning of transceiver performance. The proposed physical-layer is combined to the SDN paradigm in order to realize an efficient wavelength routing by performing both rate adaptation and node synchronization within the data center. The demonstrated results show that adaptation speeds reach a minimum of $170\ ms$ for transceivers using Pulse Amplitude Modulation (PAM) and Low-Density Parity-Check (LDPC) coding. The authors show also an improvement of 9.3% in system capacity comparing to fixed modulation and coding scenario. Besides, the authors in [157] aim to improve the scalability and flexibility of SDN-based hybrid optical DCN by minimizing the latency within circuit switching at the core layer and enhance the rapidity of packet switching at the aggregation layer. In fact, the authors sights seeks on improving the communication between fully optical connected servers for an inter-data center and intra-data center traffic. To do so, the authors consider high-capacity Optical Circuit Switching (OCS) nodes and large scalable Optical Packet Switching (OPS) to build the proposed architecture. Indeed, OCS nodes are interconnected through a mesh topology forming the clusters where each cluster is connected to two different OCS nodes. In each cluster, racks are interconnected via the OPS nodes. Severs in the same cluster are using multiplexing optical communication via the OPS nodes. Results show that the proposed routing algorithm allows a stable packet loss probability equal to 0.16% for a failure up to 15%. In [158],

the authors aim to enhance the optical spectral consumption. In fact, the proposal consists on minimizing the spectrum usage and operation complexity by rearranging the multicast-trees. The solution was implemented in inter-data center environment by choosing the path dynamically. Finally, in [159], the authors propose an SDN application prototype for Optical DCNS that called A-SMART. The proposal aims to enhance the performance of the network in case of defragmentation engendered by the occupation of the sub-optimal spectrum. A-SMART proposes a re-routing strategy in case of optical connection failure due to the physical degradation of the optical medium. Unfortunately, the above presented proposals need to be studied in term of cost, implementation complexity and energy consumption for a better DCNs performance and QoE of end users.

Despite the benefits provided by the aforementioned solution tackling the SDN network management issues in wireless DCNs, these areas of studies still need more enhancement especially to support fast client mobility when connecting to the antennas within wide DCNs. In fact, in this case, the wireless networks still require a higher reliability and reachability.

3.6 Data Center Routing protocol Comparison

We summarize the aforementioned routing algorithms in the Tables 3.1 and 3.2 through different criteria:

- **Approach name**: the name of the proposed strategy scheme.
- **DCN architecture**: defines where the routing algorithms are implemented and tested in the DCN.
- **Solution formulation**: precises the deployed solver for the optimization proposed algorithm (heuristic, ILP, etc.).
- **SDN traffic control:** defines if the proposed architecture is centralized by using SDN controller for management.
- **Routing mode**: details the communication mode deployed in the implemented proposal (online unicast/multicast or batch mode).
- **Contribution objectives** of each proposal.
- **Evaluation parameters**: details the different metrics, criteria and/or parameters used to evaluate the performance of the designed strategy.

Table 3.1 – Comparative summary of literature on routing and resource allocation in DCNs (1/2)

Ref.	Year	Approach	DCN	Solution Formulation	SDN Traffic Control	Routing mode	Contribution Objective(s)	Evaluation Parameter(s)
[44]	2010	Symbiotic Routing	CamCube	N/A	No	Online Unicast	Allowing adaptive Routing for applications	CPU utilization
[112] [111]	2013 2014	Key-based Routing	3D Torus	N/A	No	Online Unicast	Supporting developing & Running Key based servers	Throughput Resilience
[114]	2017	MPCC	EO-DCNs	ILP Heuristic	No	Online Unicast	Enhance the spectrum efficiency & System survivability	Blocking probability spectrum consumption
[113]	2017	-	Optical DCNs	K-shortest path Algorithm	No	Online Unicast	Demonstrate of trade-off between network performance & transceiver design	Blocking Probability
[109]	2020	Fault-free & Fault-tolerant Routing	LaScaDa	N/A	No	Online Unicast	Propose LaScaDa topology Enhance hybrid-DCNs performance	Bottleneck Throughput Topology Diameter Bisection Bandwidth Path Length
[105]	2020	TCAM	Multi-tiered DCNs	N/A	No	Online Unicast	Enhance TOR/leaf switches performance Mitigate critical problems of high scaled DCNs	Latency Throughput
[108]	2018	BFRouting	BCDC	N/A	No	Online Unicast	Studying BCDC construction & performance	Path Failure Throughput topology diameter
[127]	2018	DMDP	Switch only DCNs	N/A	Yes	Online Multicast	Fast discovery of disjointed multiple paths Improve traffic separation	Effectiveness ratio
[110]	2020	BRFT Routing	BCDC	-N/A	No	Online Unicast Online Multicast Broadcast	Studying the BCDC performance & deployment complexity	node-disjoint paths average CPU transmission time data transfer rate
[115]	2018	PRRO	Hybrid Wireless DCNs	non-MILP Heuristic	No	Online Unicast/ Multicast	Reduce the sum of content storage & content downloads costs	cost
[116]	2020	XSR	Typical DCNs	non-MILP Heuristic	No	Online Unicast/ Multicast	Reduce the forwarding router operations complexity	Path length Path Label size
[124]	2019	D2B	Fat tree	single-leader-multiple-follower Stackelberg game	No	Online Multicast	Maximize the network throughput decrease the network delay based on IoT devices	Bandwidth Utilization Network Delay
[122]	2020	-	Leaf Spine DCNs	ILP	No	Online Multicast	Maximize the traffic load	Accepted requests ratio Accepted requests throughput Load on fabric Uplinks
[120]	2018	DiRP	Clos	ILP	No	Online Multicast	Minimize the path setup time distributed decision for available bandwidth	Accepted requests ratio Accepted requests throughput Load on fabric Uplinks
[119]	2019	iRP LiRP	Clos	ILP	Yes	Online Multicast	Enhance the Multicast tree construction Improve the DCN capacity	Active Flow Distribution Density Accuracy of neural algorithms
[118]	2017	iRP	Clos	N/A	No	Online Multicast	Maximizing the bandwidth availability & Multicast flows distribution	Number of tree branches Link utilization capacity of available links
[134]	2017	CMRSA DMRSA CMRSA/DMRSA	EON DCNs	ILP Heuristic	No	Online Multicast	Minimize congestion Enhance spectrum efficiency	Average Occupied Spectrum Average Modulation Level Blocking Probability
[132]	2019	PDSLT-SP	EON-DCNs	ILP Heuristic	No	Online Multicast	Protection of multicast communication from Link Failure Provide high spectrum efficiency	Spectrum Resource Utilization Blocking Probability
[135]	2017	-	Optical DCNs	Monte Carlo simulations	No	Online Multicast	Scalable and modular multicast routing Support data center locality Avoid long physical paths Enable spatial wavelength reuse Support traffic flows with several requirements	AWG crosstalk Throughput
[129]	2018	NEMO-Group NEMO-cluster	Hybrid Wireless DCNs	N/A	No	Online Multicast	Save the multicast traffic Improve Throughput of DCN Enhance multicast tree quality	Congestion control Throughput CDF of Multicast Group

Table 3.2 – Comparative summary of literature on routing and resource allocation in DCNs (2/2)

Ref.	Year	Approach	DCN	Solution Formulation	SDN Traffic Control	Routing mode	Contribution Objective(s)	Evaluation Parameter(s)
[131]	2018	DST-OM MSST	EON DCNs	ILP Heuristic	No	Online Multicast	Reduce the total spectrum consumption of all multicast demands	Spectrum consumption Blocking probability
[143] [142]	2017	JRBC-HDCN SJB-HDCN	Hybrid wireless DCNs	heuristic Lagrangian relaxation	No	Batch mode	enhancing throughput reducing interference effects	link state Bandwidth Throughput
[158]	2017	Flow Scheduling	Hybrid wireless DCNs	heuristic	No	Batch mode	Enhancing throughput Reducing interference effects	Link state Bandwidth Throughput
[143] [142]	2017	JRBC-HDCN SJB-HDCN	Hybrid wireless DCNs	heuristic Lagrangian relaxation	No	Batch mode	enhancing throughput reducing interference effects	link state Bandwidth Throughput
[158]	2017	FCTcon	Fat tree DCN Elastic tree	Linear model	No	Batch mode	Dynamically control the FCT of delay-sensitive traffic flows	Traffic Load FCT Power saving
[141]	2019	TIO BFS	Hybrid Wireless DCNs	Linear Programming	No	Batch mode	Minimize network congestion Balance traffic load	Latency Packet Loss rate Throughput Routing complexity
[138]	2017	CEDM	Leaf-Spine topology	N/A	No	Batch mode	Minimize throughput loss Enhance flow completion time	Throughput Flow Completion Time
[137]	2016	ARS	Clos	N/A	No	Batch mode	avoid incast TCP problem caused by highly concurrent flows	Good put RTO event Packet Loss
[139]	2019	ExpREsS DVFS	inter data center	EDF	No	Batch mode	Enhance the orchestration of Spark applications & Decrease energy consumption	Power consumption applications deadlines
[160]	2016	MCDC	Fat Tree	N/A	Yes	Online Multicast	Minimizing the latency Leveraging congestion control	Congestion control Scalability Security
[146]	2018	ECMP	Fat Tree	Modulo-n hashing method & Modified Djikstra	Yes	Online Unicast	enhance TCP and UDP performance Better path selection	Throughput Jitter Packet Loss
[145]	2020	cRetor	Fat Tree/DCell	Heuristic	Yes	Online Unicast	Speed up the routing calculation efficient topology description	Path computation performance network convergence speed fault recovery capability routing overheads
[147]	2017	NLADR	Hierarchical DCNs	N/A	Yes	Online Unicast	Minimizing the congestion Improving Throughput	Number of Arrived flows Throughput
[150]	2020	DQL	Fat Tree	Reinforcement learning (RL)	Yes	Online Unicast	Reduce latency and packet loss & Increase throughput for mice/elephant flows	Average delay Throughput Packet Loss Rate
[148]	2018	L2RM	Fat tree	N/A	Yes	Online Unicast	Improve the performance of Fat Tree DCNs Avoid Links congestion	Link utilization message overhead average usage rate of flow tables
[157]	2019	-	Hybrid Optical DCNs	N/A	Yes	Online multicast /Unicast	Enhance the scalability & the flexibility of the network	Packet Loss
[154]	2018	Rule reduction	Hybrid Wireless DCNs	NP hard Problem Heuristic	Yes	Online Unicast	Reduce the number of flow setup requests generated by the switches over wireless DCN	Number of installed rules Number of flows setup requests
[149]	2020	PAM	Clos	ILP	Yes	Online Multicast	Minimize the average multicast completion times	convergence time scalability Link capacity multicast fanout
[155]	2019	greedy- heuristic	Hybrid Wireless DCNs	Heuristic	Yes	Online Unicast	Reduce total completion time for network updates	convergence time scalability Link capacity multicast fanout
[158]	2017	-	Hybrid Optical DCNs	N/A	Yes	Online Multicast	Improvement of optical spectral consumption	Bandwidth Delay
[159]	2020	A-SMART	Optical DCNs	N/A	Yes	Online Unicast	Reconfiguration of Optical network Re-routing in case of connection failure	optical connection failure

3.7 Conclusion

In this chapter, we presented the different proposed routing schemes existing in the literature for the Typical DCNs and the Enhanced DCNs. Besides, we detailed the most relevant strategies propounded in the literature tackling the SDN based data center routing. In the next section, we will describe the implemented and deployed experimental platform to test and evaluate our proposed contributions in this thesis.

Chapter 4
SDN-based CamCube platform: Implementation & Deployment

Contents

4.1 Introduction

Recently, the virtualization techniques and Software-Defined Network (SDN) have changed the manner of creating and managing cloud based applications. In fact, thanks to the SDN mechanism, service providers can due to the real-time monitoring of the substrate network in terms network topology and available residual resources. Also, in order to ensure the required QoS level for these cloud applications (e.g., bandwidth, delay, packet loss and jitter), an optimized resources allocation algorithms should be implemented and incorporated in the SDN controller. Before doing so, we suggest to start by describing SDN network architecture.

In fact, the distributed systems (traditional networks) are self-organized. The control plane of each device in a traditional scheme implements distinct network functions. In addition, each of these functions implements a mechanism to distribute and synchronize state across the network (e.g., local view of the topology and link state). Besides, we need a standard protocol to allow devices from different vendors communicating and treating these control messages. However, each function needs to implement a similar mechanism for maintaining a given state, and thus an important volume of code found on every device's framework. Also, these traditional networks are hard to configure and to operate correctly especially when they are positioned in different locations. Moreover, it is hard to troubleshoot and to update them due to the absence of a single macro view of the system (each device builds its own view of the topology based on local information). Consequently, following the SDN scheme, we can abstract networks in two separate operating planes: data and control planes. On one side, the data plane implements the basic forwarding functionalities i.e., we can represent the data plane operations as a function of the packet header and metadata which results in an output port or a drop. Indeed, the data plane is usually abstracted with forwarding tables. On the other sides, the logically centralized control plane is responsible for computing the configuration of each physical or virtual device in the network. Control plane handles the populating of the forwarding tables of the data plane, depending on the implemented function, such as routing, isolation or traffic engineering. The controller has a global view of the network topology and the resources use retrieved by the data management plane. Therefore, this control plane is usually abstracted as functions over the global topology graph and weights e.g., to produce a routing table or to balance load. Thereby, we can implement powerful optimization algorithms (e.g., functions) in the centralized control plane to enhance network performance, satisfaction of QoS, and data center reliability.

Moreover, SDN illustrates an important role to facilitate the network resources management and to efficiently virtualize the DCNs in an on demand manner [161][162]. In fact, the SDN technology can easily grasp the real-time network state information and allocate the network resource according to cloud services requirements by using the controller Northbound and Southbound APIs. In fact, someone may be wondering if the SDN based routing performs well in server-only DCN. In order to answer this question, we propose a novel optimization SDN-based CamCube DCN algorithms that can be analyzed and evaluated through a realistic experimental platform scenario detailed in this chapter. To that end, we will see in the next sections of this chapter how we deploy the CamCube DCN built in the traditional networking scheme (explained in Chapter 2 in Section 2.2.2.2) and SDN based scheme (i.e., in a logically centralized scheme).

4.2 Proposed Architecture

On the opposite to the original design of CamCube, we propose a logically-centralized control plane (i.e., by deploying SDN controller). We make use of Open VSwitch (OVS) to communicate among CamCube servers instead of conventional virtual switching provided with hypervisors in order to ensure network monitoring and control. In fact, all these forwarding elements (i.e., Open vSwitch devices) implemented in CamCube servers are directly connected through a secure socket to the SDN controller ONOS and report the network status as an ordinary SDN switching device via the SouthBound Interface (SBI) APIs of ONOS, e.g., OpenFlow. Then, the latter is able to receive real time information about the whole managed topology. Besides, in our design, we propose an orchestration management system where it accepts as an input the whole network information provided by ONOS through a NorthBound Interface (NBI). This orchestration system communicates with the external applications of ONOS, i.e., via Restful APIs, in order to collect these statistics compulsory for the calculation of the optimized routing path. Indeed, the orchestration system contains a solver module that is responsible for routing the arrived flows with the required resources such as the bandwidth. Consequently, according to the considered state of network and pre-defined constraints, the solver proceeds to the calculation of optimized paths between any given source and destination. Once the solution of our proposal is set, i.e., the optimized path is provided by the orchestration system, the ONOS controller installs the calculated path via flow rules through its SBI in CamCube by using the corresponding drivers and modules. Hence, in this case, ONOS uses the OpenFlow drivers to install packet forwarding rule and build the topology depending on the state of the network requests in terms of: topology, switches, links, hosts, residual resources, etc. Figure 4.1 illustrates our global SDN based CamCube architecture and shows the interaction between its main elements as described below.

The remainder of this chapter is organized as follows. Section 4.3 will resume the ONOS SDN controller. In particular, we will start by introducing ONOS core subsystem structure, the network programming features and a flow abstraction view will be then debriefed. Afterwards, we will detail the key northbound, southbound and flow objective abstractions. Also, we will discuss the existing three ways for interacting with ONOS. Next, we will enumerate the steps of implementation and deployment of the proposed solution in Section 4.4. In Section 4.5, we will list the several performance metrics that we will consider for analyzing the performance of the proposed solution. Finally, Section 4.6 will conclude the chapter.

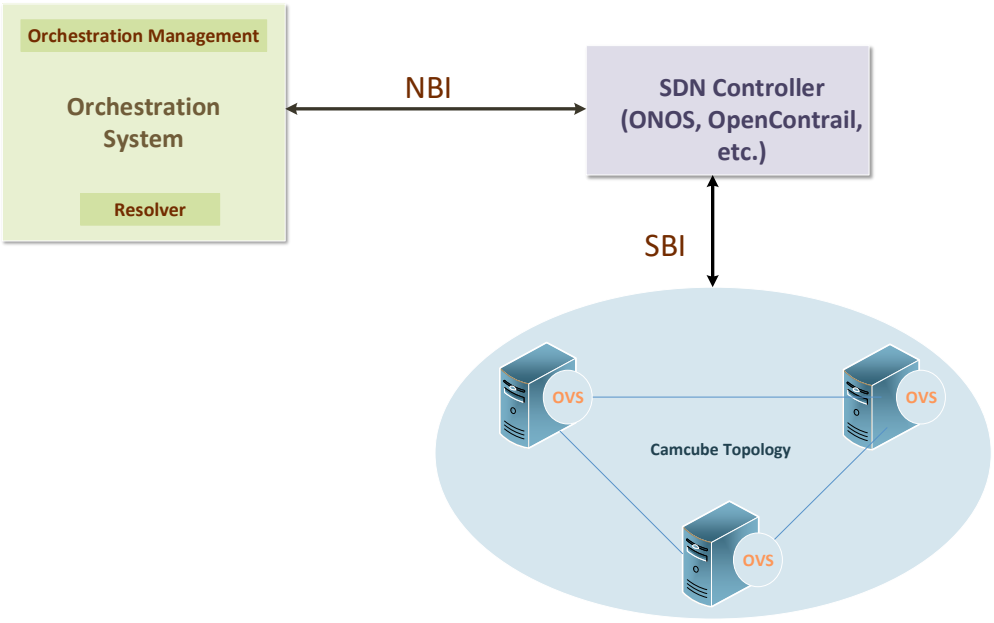

Figure 4.1 – Global view of the platform

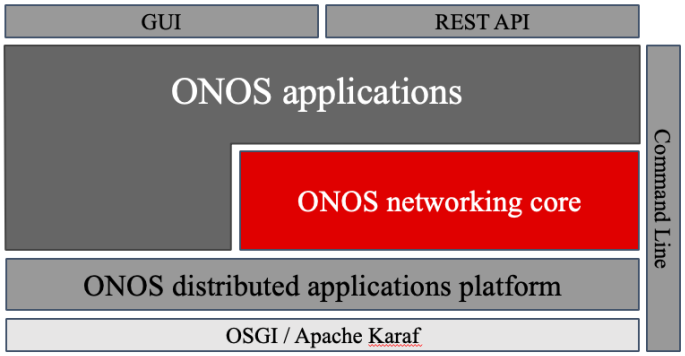

Figure 4.2 – ONOS Structure [163]

4.3 SDN Controller: ONOS

ONOS SDN controller is an open-source project of the Open Networking Foundation (ONF) [71]. This SDN controller is characterized by: i) a Command Line Interface (CLI) and a Graphic User Interface (GUI) in order to easily managing applications and features, getting information about network devices and setting configuration, and ii) it is also largely deployed to control the infrastructure of operators and service providers such as: Google and AT&T. Moreover, ONOS has a very active user community and well documented. Recent studies show that this controller offers a better performance in comparison with its main concurrent products such as Open-

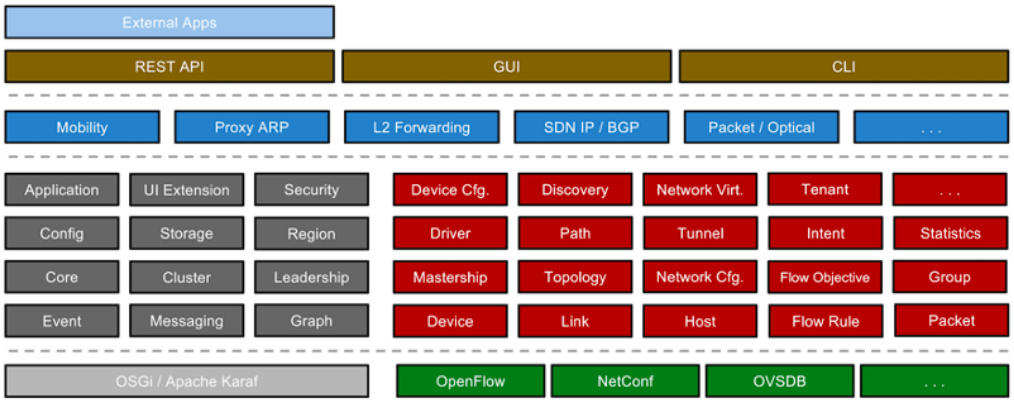

Figure 4.3 – ONOS Subsystems overview [164]

Daylight [165] and Ryu [166] in terms of bandwidth [167] and latency [168, 169]. It is shown that the control plane traffic for an ONOS cluster (i.e., 3 to 7 ONOS controllers) is negligible with respect to the data plane traffic. In fact, in 2017, ONF has created a special working group from industrial and academic partners to asses the different performance issues of distributed ONOS so-called "Security & performance analysis brigade" [170] and where the group publishes an annual performance report. The aforementioned advantages motivate us to deploy this SDN controller in the design of our proposed architecture. To manage the network, the controller uses its SouthBound Interface (SBI) providing several protocols such as OpenFlow, NetConf, XMPP in order to install the flow rules (e.g., OpenFlow rules).

ONOS is a multi-modules controller in which modules are managed as Java OSGi bundles as shown in Figure 4.2. It is designed to be modular system with protocol agnosticism and separation of concern principals. Indeed, it is based on clear boundaries between subsystems to facilitate modularity as shown in Figure 4.3. Last, we will make use of the network programming abstraction to make our deployment independent of the under-laying topology (e.g., driver of devices). Let us explain here the ONOS point of view of this abstraction [71]. As shown in Figure 4.4, the devices are first abstracted via flow rules to program a forwarding table, such as OpenFlow, NETCONF, and XMPP. Forwarding tables and flow rules provide a portability between the SBIs (control or configuration protocols). This abstraction enables application developers to avoid the multitude ways to accomplish similar objectives by each SBI and the difference in fields between versions of the same SBI protocol. Network applications are built for one or more protocols are effectively locked into a specific set of devices or vendors, and hence tend to increase in complexity as they become more universal. In fact, many forwarding devices expose more than one table, and each one is individually programmable through the Flow Rule service. It is quite common that each table has different restrictions and capabilities about which headers can be matched and transitions are allowed. Thus, the applications need to be customized into the tables and pipelines of each individual device. For instance, this task is not easily tractable with large topology as the CamCube DCN. Standards organizations, like ONF, have tried to address this issue by defining a generic ONOS pipeline that gets mapped to different tables through pipeline drivers via the second level of abstraction (Flow objective) that we

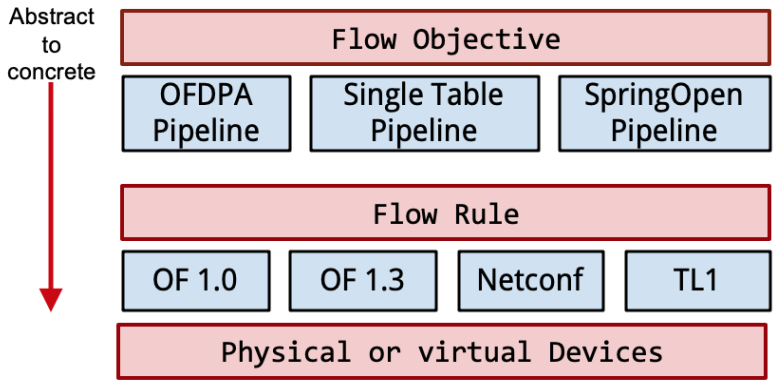

Figure 4.4 – Network programming abstraction levels in ONOS [171]

have used in our platform.

An important piece of the SDN controller architecture gets along with the development of OpenFlow protocol [152] as one of the main SBIs today. In fact, this configuration protocol helps to i) create a separate networks only for network control functions, ii) to establish the network programmability, and iii) ensuring logically centralized control. Therefore, using the flow objective abstraction through OpenFlow SBI, we are able to insert, delete, modify and lookup for each entry in the flow tables of the available devices. Note that in the flow table of an Openflow switch, each entry is characterized by i) `headers` to match the received packets, ii) `action` to specify what to do for the matched packets and iii) `statistics` of the matched traffic.

4.4 Implementation & Deployment of the Proposed Architecture

In order to implement the proposed solution, we have developed a python script that we named "emulation_camcube" in which we detail the emulated functionality of our orchestration system and define the inputs and outputs of each module to establish the desired interaction between the different elements, i.e., fully virtualized CamCube topology, ONOS controller and the proposed orchestration system. Hereafter, we specify the implemented functions and deployed platform of our proposal.

4.4.1 Implementation

Figure 4.6 shows in detail the call graph of the *"scheduler_loop"* function. This function calls the *"flow_generation"* function that is responsible for the generation of the arrived flows in the system according to a defined flow distribution, throughput and some required resource bandwidth that simulate the user request and an input of the system. The function *"scheduler_loop"* calls the *"flow_process"* function responsible of the scheduling of the arrived flows in the system. This function provides the whole information related to the processing of the flow from its arrival till its departure from the system. In fact, it includes the processing time spent by a

solver (e.g., Cplex or a heuristic algorithm) to find a solution, mainly an appropriate path from a source to a destination with respect to some constraints, in addition to transmission and propagation response time. Indeed, the *"flow_process"* function calls the following functions as shown in Figure 4.6:

- *"configure_path"* takes as inputs each flow characteristics such as bandwidth, source, destination, traffic profile (e.g., UDP), etc. Then, sends it to the ONOS controller in order to install the flow rules according to the resulted optimized path obtained by the *"install_flow_rule"* function. Thus, the output of this function is the generation of the flow rules of each resolved flow by our system.

- *"solve_for_path"* function that details the role of the solver in our proposal. In fact, in this function, we define either our objective function and its constraints or a heuristic algorithm. The output of this function consists on the optimized path between randomly selected source and destination(s) of each admitted flow. The obtained path is sent from the orchestration system to ONOS via the Restful API in order to be installed later by the SDN controller into the OVSs located inside the CamCube servers.

- *"launch_traffic_generator"* is responsible for the traffic generation using "iperf" tool between CamCube source and destination nodes of each admitted flow once the corresponding path is installed. In particular, we configure the iperf client side here. The output of this command is the report of the iperf traffic generated between the source and destination of the flow.

- *"launch_traffic_listener"* function defines the command of iperf in the server side. In this function a CamCube node plays the role of a server that receives packets from a corresponding client according to a predefined traffic profile of a particular flow, such as: bandwidth request, duration of the transmission, type of the transport protocol, etc.

- Note that each flow is characterized by a process ID in the system i.e., pid. We implemented a tracker function, within the *"flow_process"* function, able to estimate the residual bandwidth and to maintain the right rules in the devices. Each flow has the *"is_pid_running"* function to identify the time to live from the system. Consequently, upon finishing the transmission for a flow, the rules should be cleaned from the data plane and confirmed by *"try_kill"* function. Besides, concerning the estimation of the current used bandwidth, it can be calculated in a real SDN application thanks to the (org.onosproject.net.statistic.StatisticService) interfaces [172]. The latter interfaces are able to provide a periodic statistics of sent bytes per port thanks to the OFPT_MULTIPART_REPLY, OFMP_PORT_STATS messages information and hence the throughput as shown in Figure 4.5. This Figure depicts a captured openflow packets e.g., the frame number 32 shows a OFPT_MULTIPART_REPLY message sent by a switch to the ONOS controller. In this reply message, we can notice that port number one received 28 packets with a total volume of 44269 bytes and on the other direction, the same port sent 339 packets with a total volume of 2088 bytes.

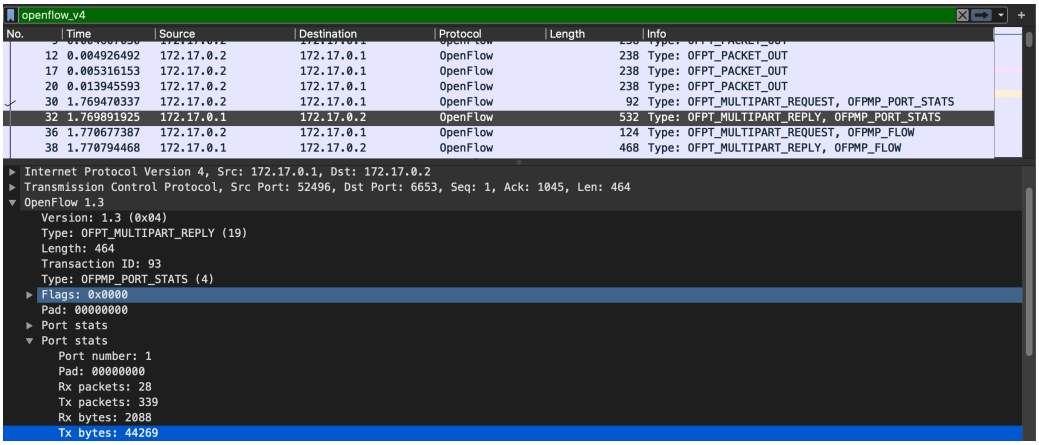

Figure 4.5 – Capture of Openflow statistics and messages

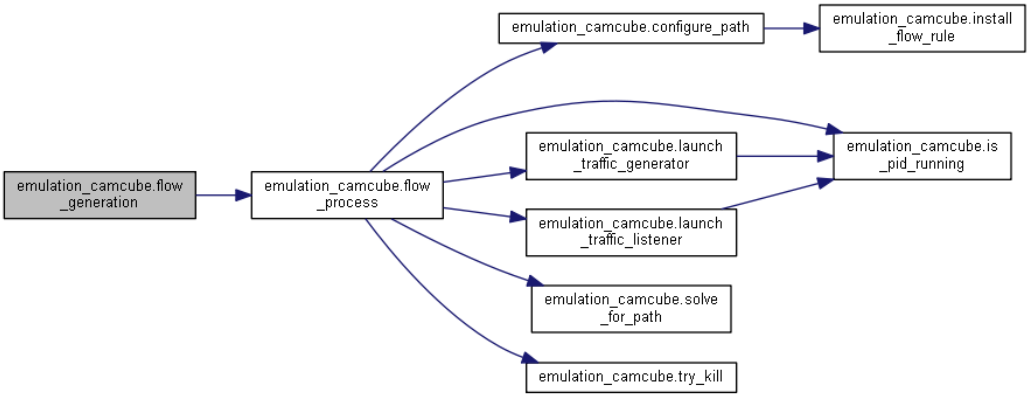

Figure 4.6 – The call graph of scheduler_loop function

4.4.2 Deployment

In this section, we define the essential tools used in the deployed platform.

4.4.2.1 Platform & Tools

- Mininet: To emulate the CamCube topology, we propose to make use of Mininet [173] [174]. It is an open source network emulator developed to support research in SDN paradigm. It creates a network of virtual hosts, links and switches and can be associated with a remote controller such as ONOS. In this emulator, we make use of Mininet hosts to simulate the CamCube servers and we ensure the switching function by using the OVS deployed and configured in Mininet.

- Myiperf: We propose to use "iperf" tool in order to generate the traffic of each arrived flow between source and destination(s) and further to evaluate the quality of service in terms

of latency, loss and jitter. However, to the best of our knowledge, we noticed that it doesn't exist a version of an iperf tool that is able to generate the statistics related to the latency for UDP flows. Consequently, we propose a new version of iperf that we named ***"Myiperf"***. In the latter version, we aim to modify and adapt the source code to our needs by adding the different functions related to the latency statistics. Note that the corresponding code is uploaded in the GitLab of Devoteam, accessible via the private network of the company.

- Docker: We install docker in our platform in order to use the dockerized ONOS SDN controller. It simplifies the deployment of ONOS cluster [175]. Note that in our experiments, we do not make use of clustering feature (i.e., using logically centralized ONOS with several distributed controllers). In fact, with a cluster, we can ensure the reliability and overcome the single failure point of the controller. However, using a cluster will increase slightly the communication load between the controllers to maintain a consistent state in the network. According to ONOS performance report [176], the impact of the communication overload between ONOS nodes on the overall performance is negligible. Note that high number of ONOS nodes implies more consumed CPU/MEM/BW resources of our experimental server, and for that reason we just used a single ONOS node.
- Python: We make use of python programming language [177] to develop and implement the different modules and functionalities of our proposal.
- Cplex: To resolve the Linear Programming problem, we propose to use Cplex solver. To do so, we implement the optimization algorithm by using the Cplex python API proposed by IBM [178].
- Server: In order to emulate and deploy the above platform, we make use of a powerful bare-metal server characterized by 515 Go of RAM, 56 cores CPU, frequency equals to 2600.026 MHz and 2.7 To of local disk. Note that we deployed the Ubuntu 18.04 sever as operating system in this sever.

4.4.2.2 Validation of the Platform

Once we installed and configured the above mentioned modules, we interconnect ONOS to emulated CamCube topology. As a result, we can list the devices, links, and hosts from Karaf or visualize the full virtualized CamCube in the ONOS GUI as shown in Figure 4.7. In this example, we emulate 8 servers sized CamCube topology in Mininet. Note that each pair of host and direct connected OVS Switch simulates a bare-metal server We can simulate the links (cables) distances and their capacity through Mininet API in Python. To sum up, in our proposal model, a CamCube server is modeled by i) OVS switch to ensure the forwarding role between the CamCube nodes and ii) a network namespace host for computation (e.g., run services and applications). Then, the emulated CamCube node is visualized in the GUI of ONOS as shown in Figure 4.7.
Our proposed design is operational and will help us to validate or not any optimization solution in realistic settings. Hereafter, we define the different parameters of the proposed scenarios used in the experimentations, mainly the traffic profile (i.e., the distribution of the requests and the flow characteristics).

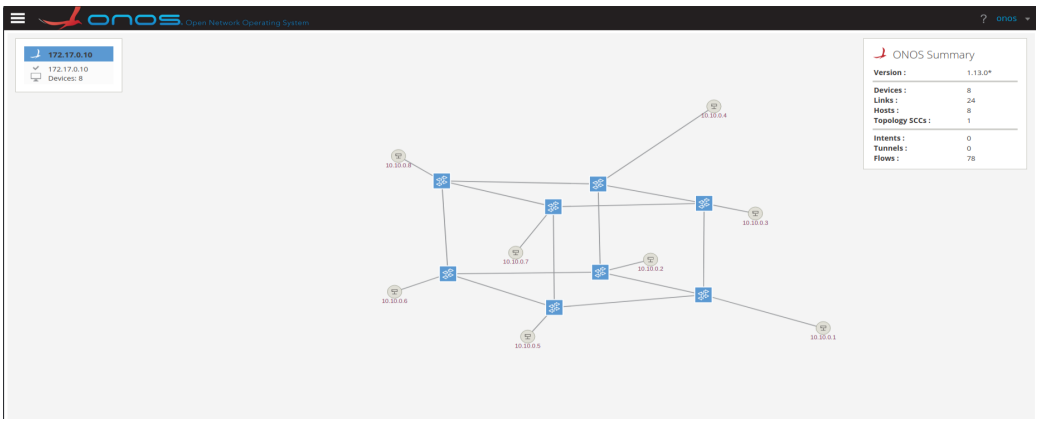

Figure 4.7 – Display of CamCube servers in ONOS

4.4.2.3 Experiments settings

- ***Setup for CamCube Topology:***

  This section summarizes the experiments settings of three main routing traffic scenarios: i) Unicast, ii) Multicast and iii) Batch. To assess the scalability issue we vary our DCN topology size from $5 \times 4 \times 5$ (i.e., 100 servers) to $5 \times 4 \times 12$ (i.e., 240 servers). We consider symmetric links between neighboring servers with maximum capacity of $100\mathrm{Mbps}$.
  To simulate application traffic, we assume total number of 100 flows. The arrival rate of each flow follows a Poisson process with density $\lambda_f = 12$ flows per second for most experiments. It will be mentioned when considering a different density. The selection of the source and destination(s) hosts follows a random uniform distribution among all CamCube servers. Also, we fix the size of multicast destination group following uniform random distribution between 2 and 10.

  The duration of flows follows the exponential distribution with average of $\lambda_d = 180\mathrm{s}$. If an arrived flow cannot be admitted by one of our proposals due to network congestion (i.e., not sufficient resources), the given traffic flow should wait in a particular queue until succeeds its treatment in the network. Notice that each flow attempts to be treated by our solver each Δ_t time. In our case, we fixed $\Delta_t = 1sec$ considered as the frequency of resolution attempts. Consequently and for an in-depth evaluation, we define $\mathcal{A}$ as the average number of attempts for all arrived flows to be admitted and resolved by the proposed solution. Besides, we define $\Delta_b = 1\mathrm{sec}$ as a batch window when we study the batch-routed flows detailed in chapter 7.
  Based on the configuration of a real Cisco router setup [179], we set IP queue size in CamCube servers to 20000 packets. Moreover, we set the packet size to 1500 bytes [180] and we generate UDP/TCP flows. The throughput of Constant Bit Rate (CBR) traffic follows uniform random distribution between 20 and 30 Mbps. We make use of iPerf tool to generate UDP and/or TCP based traffic flows. The performance results are calculated with confidence intervals equal to 95%. Hereafter, we summarize the aforementioned inputs in Table 4.1 below.

Table 4.1 – Summary of scenario for CamCube topology

Parameters/Tools Name	**Value**
Topology Size	100 – 240 servers
Number of flows	100
λ_f	12 flows/second
Δ_t	1 second
IP queue size	20000 packets
Traffic	UDP or TCP for the transport layer
Multicast destination group of hosts	[2 - 10] CamCube servers
λ_d	180 seconds
The packet size	1500 bytes
Throughput	[20 - 30] Mbps
Δ_b	1 second
Confidence Intervals	95%
Flow generator tool	myiperf
Emulator	Mininet

- ***Setup for Clos Topology:***

  To evaluate the performance of our proposals, we compare the obtained results in CamCube with Clos topology. In this section, we detail the experiment setup for this multi-hierarchical topology. First, in [181], the authors consider two-leveled Clos topology with 64-port switches, we can connect 2048 servers. Note that in our simulations, we can reach only at maximum 240 sized topologies due to the software limitations, we implement a two-leveled Clos topology for test. We assume that our spine and leaf/ToR switches are the same n-port bandwidth capacities. Then, we fix the number of servers in each rack to 12 hosts according to the Huawei setup [182]. Besides, we define the bandwidth of ToR switches to 100 $Mbps$ and the links capacity between ToR and Leaf switches to 100 $Mbps$. Finally, we set the links capacity between Leaf and Spine switches to 400 $Mbps$. This is to simulate the 10/40 $Gbps$ in real scenario. It is noteworthy to say that the maximum capacity of links configured in Mininet is set to 1 $Gbps$ which explains the choice of the link capacity value for the spine switches within the tested Clos topology. Table 4.2 summarizes the aforementioned setup.

In Table 4.3, we show the number of the different network equipment to build the topology of the platform according to the number of servers.

Table 4.2 – Summary of scenario for Clos Topology

Parameters/Tools Name	**Value**
Topology Size	100 – 240 servers
Number of flows	100
λ_f	12 flows/second
IP queue size	20000 packets
Traffic	UDP or TCP for the transport layer
Multicast destination group of hosts	[2 - 10] CamCube servers
λ_d	180 seconds
The packet size	1500 bytes
Throughput	[20 - 30] Mbps
ToR Links	100Mbps
ToR-to-Leaf Links	100Mbps
Leaf-to-Spine Links	400Mbps
Number of servers per Rack	12servers
Confidence Intervals	95%
Flow generator tool	myiperf
Emulator	Mininet

Table 4.3 – Summary of the Clos Topology Infrastructure

Number of servers	**Number of racks**	**Number of ToR switches**	**Number of Spine switches**
100	8	8	5
120	10	10	6
140	11	11	6
160	13	13	7
180	15	15	8
200	16	16	9
220	18	18	10
240	20	20	11

4.5 Performance metrics

In this section, we define the several performance metrics that we considered for analyzing the quality of service of the generated UDP and/or TCP traffic in our platform for our three scenarios: unicast, multicast and batch-routed flows.

- *Packet Loss Rate:* We define the Packet Loss Rate ($\mathbb{P}$) as the percentage of IP packets lost for the total number of flows.
- *Latency:* We consider the all finished transmitted flows in the network and we calculate the latency ($\mathbb{L}$) as their total packet average delay. For a single flow $\mathcal{F}_i$, we define l_i as the average delay for its total transmitted packets. N is the number of all finished flows in the network. Then, $\mathbb{L} = 1/N \sum_{i=1}^{N} l_i$
- *Convergence time:* It calculates the time that an optimization algorithm takes to provide the path solution for a given flow. We denoted the convergence time by $\mathbb{C}$.
- *End to End (E2E) Delay time:* Defines the average time of the treatment of an accepted flow by our system. Hence, the end to end delay time, named $\mathbb{E}$, includes the sum of the convergence time $\mathbb{C}_i$ in addition to the transmission and propagation time $\mathbb{L}_i$. Then, we define $\mathbb{E}$ as: $\mathbb{E} = 1/N \sum_{i=1}^{N} e_i$ where N is the number of all finished flows in the network and $e_i = \mathbb{C}_i + \mathbb{L}_i$.
- *Jitter:* We define $\mathbb{J}$ as the variation of the latency for all transmitted flows within CamCube DCN.
- *Path Length:* $\mathbb{P}_a$ denotes the average length of the path used between sources and destinations during the experimentation for all flows. Let p_i denote the traveled hop of flow $\mathcal{F}_i$ and N the total number of finished flows in the network. Then, $\mathbb{P}_a = 1/N \sum p_i$.
- *Relative-Error:* We make use of $\mathbb{RE}$ ratio in order to evaluate and explain the variation of the behavior of the studied routing schemes according to a specific performance metric. For example, we calculate $\mathbb{RE}$ ratio of the shortest path routing in CamCube and in Clos DCNs in order to compare it to the optimal solution presented by our proposal in term of packet loss within 240 servers sized topology. We calculate $\mathbb{RE}$ as follow : $\mathbb{RE}(x,y) = \frac{\mathbb{RE}(x)-\mathbb{RE}(y)}{\mathbb{RE}(x)}$

  where each x and y represent the performance metrics value for a considered routing protocol within a predefined topology. Note that if x provides the optimal solution, when $\mathbb{RE}$(x,y) ratio increases then y moves away from the optimal solution x.

For the multicast flows, in addition to the aforementioned performance metrics related to the QoS of the UDP traffic, we consider the quality of the generated multicast tree and we aim to compare the multicast solution resulted from our optimization algorithms to the results generated by the shortest path algorithm. To do so, we consider:

- *Number of Resolution Attempts:* If the arrived flows cannot be admitted by our proposal due to network congestion (i.e., not sufficient resources), the multicast traffic flows wait in the queue until the algorithm succeeds its deployment in the network.
- *Full Tree Per Multicast Group:* This metric quantifies the average ratio between the size of i) full multicast tree nodes and ii) multicast group. Note that if this metric is large means that many relay nodes have been used to build the multicast tree. $\mathbb{T}_i$ denotes this ratio for flow F_i. Thus, $\mathbb{T} = 1/N \sum \mathbb{T}_i$

Considering TCP flows, we evaluate the performance of our proposal through the *Throughput* performance metric obtained from Iperf server report. The latter designs the maximum quantity of transferred data for a flow during its prefixed duration of transmission in the network i.e., λ_d.

4.6 Conclusion

In this chapter, we have introduced the implementation details of the platform that we will use to evaluate any SDN based routing algorithm in CamCube data center topology. We deployed several open-source projects such as ONOS, Mininet, Iperf, OpenVswitch, in addition to many python packages. Also, we have defined the traffic profiles and the performance metrics used later in the next chapters in the experiments/results sections. It is worth mentioning that before arriving to the right technologies choice to build our plate-form, we had explored several other tools and systems like OpenContrail [183] SDN Controller, SONA project, OpenStack [184] and OpenDaylight. We have learned interesting technical skills in spite of all the deployment difficulties related to the synchronization of all these open-source projects in order to build a full common platform. In fact, some of these projects were on progress with several updated versions (up to 4-6 updates per year for some).

Chapter 5
Optimized Unicast SDN Routing Protocols in CamCube DCN

Contents

5.1 Introduction

In the recent years, large-scale data center topologies have been actively built. According to Cisco Global Cloud Index [12], hyper-scale data centers will englobe 53% of the equipment park by 2021. In fact, the forecast number of hyper-scale DCNs in 2021 is 628. The significant growth of today cloud applications (e.g., SaaS, PaaS and IaaS) leads to the congestion of DCNs and hence the quality of service may be deteriorated. According to Cisco, datacenter traffic is predicted to reach to 19.5 zetta-bytes per year by 2021 from only 6 zetta-bytes in 2016 [12]. Therefore, the optimization of traffic routing algorithms, within a DCN, plays a key role in improving the network performance and resource utilization.

In this chapter, we address first the unicast routing problem in DCNs CamCube server-only topologies. To do so, we formulate a centralized routing scheme (making benefit of the SDN) as a Mixed Integer Linear Programming (MILP) model. To solve the latter, we propose a new scheme named CamCube Routing Protocol (`CRP`) and it is based on the Branch-and-Cut algorithm. `CRP` is an SDN application deployed in the application layer of the SDN architecture on top of ONOS and it makes use of CPLEX solver as already explained in Chapter 4. Based on extensive emulations, our proposal yields better performance than the shortest-path alternative (i.e., OSPF) in terms of packet loss and latency while the path-length is slightly greater. However, the convergence time of CPLEX is not negligible mainly with large-scale DCN topology. Consequently, we propose a new resolution method named `ACO-CRP`. It shows a very good solutions for on-time routing unlike the optimum one. In order to study the efficiency of CamCube topology coupled with SDN paradigm, we will provide a full comparison of CamCube and the traditional Clos topology using the same test scenarios. We will see that CamCube is a valid option not only for distributed networks but also for the logically centralized networks (e.g., SDN-based networks).

The remainder of this chapter is organized as follow. First, in Section 5.2, we give the formal description of our routing problem of flows inside the CamCube DCN. Next, Section 5.3 details our proposed CamCube Routing Protocol (`CRP`) which is based on the Branch-and-Cut algorithm. Then, we detail in Section5.4 our `ACO-CRP` based on the ant colony optimization algorithm. Afterwards, in Section 5.5, we discuss the obtained results to evaluate ours proposals by comparing them to the existing protocols respectively in CamCube and Clos topologies. Finally, Section 5.6 concludes the chapter.

5.2 Problem Formulation

In this section, first we give the formal definition of our CamCube-based network model. Next, we formulate the path computation problem for the flows occurring within the CamCube data center.

5.2.1 CamCube Network Model

We consider the set of all connected servers designed as directed weighted graph where weights equal the link capacities. The graph is denoted by $\mathcal{G} = (\mathrm{N}(\mathcal{G}), \mathrm{L}(\mathcal{G}))$. Also, $\mathrm{N}(\mathcal{G})$ represents the set of nodes (i.e., CamCube servers) and $\mathrm{L}(\mathcal{G})$ the set of links attaching each server to neighbors.

We denote the link from n_1 to n_2 as $l_{n1}^{n2} \in L(\mathcal{G})$. The link l_{n1}^{n2} is characterized by its: i) initial bandwidth capacity $\hat{C}(l_{n1}^{n2})$, and ii) residual bandwidth $C_i(l_{n1}^{n2})$ at the time instant $\mathcal{T}_i$.

5.2.2 Centralized path computation for intra CamCube DCN flows

In this chapter, we consider the routing of flows where each flow $\mathcal{F}_i$ requests to send data with a fixed bandwidth B_i equal to $\frac{\mathcal{V}_i}{\mathcal{T}_i}$ e.g., Constant Bit Rate (CBR) flows. $\mathcal{V}_i$ is the volume of bits transferred, following a random uniform distribution, within duration $\mathcal{T}_i$ interval following the random exponential distribution. The arrival rate of $\mathcal{F}_i$ is modeled as a Poisson process characterized by λ_f. Our objective is to maximize the acceptance rate of flows in our DCN while allocating for each flow a path that maximizes the minimal residual bandwidth capacity with the minimum number of hops. The calculation and the installation of each path is performed in the SDN controller by using respectively optimization algorithms and OpenFlow rules.

In fact, the maximization of the minimal residual bandwidth can lead to an enhanced provider's revenue and respecting the Service Level Agreement (SLA) at the same time.

Let us consider the arrived flow $\mathcal{F}_i$ which is to be transmitted from source node n_s to destination node n_d. Let x_{n1}^{n2} be the binary variable which equals to 1 when the link l_{n1}^{n2} is selected as part of the path allocated for the flow $\mathcal{F}_i$. The objective can be expressed as following:

$$\textbf{maximize}\,[\min\{y_{n1}^{n2} | l_{n1}^{n2} \in L_i\}]$$

Then

$$\textbf{minimize} \sum_{l_{n1}^{n2} \in L_i} x_{n1}^{n2} \tag{5.1}$$

where L_i are all links with a capacity C_i greater than the requested bandwidth B_i. y_{n1}^{n2} is an auxiliary variable representing the residual capacity of the link after the decision, formally it equals to:

$$y_{n1}^{n2} = C_i(l_{n1}^{n2}) - B_i \cdot x_{n1}^{n2} \tag{5.2}$$

Note that only links $\{x_{n1}^{n2}\}$ with sufficient capacity C_i in terms of bandwidth can be considered to build path in our system. Formally,

$$B_i \cdot x_{n1}^{n2} \leq C_i(l_{n1}^{n2}) \qquad \forall l_{n1}^{n2} \in L(\mathcal{G}) \tag{5.3}$$

The total number of ingress flows should be equal to the egress flows for each node. Otherwise, the source node has only egress flows and the final destination node has only ingress flows.

The following constraints describe our path problem formulation:

$$\forall n_0 \in N(\mathcal{G}), \sum_{l_{n1}^{n0} \in L(\mathcal{G})} x_{n1}^{n0} - \sum_{l_{n0}^{n2} \in L(\mathcal{G})} x_{n0}^{n2} = \begin{cases} -1 & \text{If } n_0 = n_s \\ +1 & \text{If } n_0 = n_d \\ 0 & \text{Otherwise} \end{cases} \tag{5.4}$$

We consider that at most one link can be activated **into** n_0 as follow:

$$\forall n_0 \in N(\mathcal{G}), \sum_{n_1 | l_{n_1}^{n_0} \in L(\mathcal{G})} x_{n_1}^{n_0} \leq 1 \tag{5.5}$$

We consider also that at most one link can be selected **from** n_0, formally:

$$\forall n_0 \in N(\mathcal{G}), \sum_{n_2 | l_{n_0}^{n_2} \in L(\mathcal{G})} x_{n_0}^{n_2} \leq 1 \tag{5.6}$$

Our computation of the centralized path in the CamCube data center network discussed so far is summarized in Problem 1.

Problem 1 Path Computation in SDN CamCube DCN

maximize $[\min\{y_{n_1}^{n_2} | l_{n_1}^{n_2} \in L_i\}]$

Then

minimize $\sum_{l_{n_1}^{n_2} \in L_i} x_{n_1}^{n_2}$

Subject to :

$$B_i \cdot x_{n_1}^{n_2} \leq C_i(l_{n_1}^{n_2}) \qquad \forall l_{n_1}^{n_2} \in L(\mathcal{G})$$

$$\sum_{n_1 | l_{n_1}^{n_0} \in L(\mathcal{G})} x_{n_1}^{n_0} \leq 1 \qquad \forall n_0 \in N(\mathcal{G})$$

$$\sum_{n_2 | l_{n_0}^{n_2} \in L(\mathcal{G})} x_{n_0}^{n_2} \leq 1 \qquad \forall n_0 \in N(\mathcal{G})$$

$$\sum_{l_{n_1}^{n_0} \in L(\mathcal{G})} x_{n_1}^{n_0} - \sum_{l_{n_0}^{n_2} \in L(\mathcal{G})} x_{n_0}^{n_2} = \delta_{n_0}^{n_d} - \delta_{n_0}^{n_s} \quad \forall n_0 \in N(\mathcal{G})$$

$$x_{n_1}^{n_2} \in \{0, 1\}$$

In fact, the problem is a multi-objective optimization:

$$\textbf{maximize} \quad f_1(x) = \min\{y_{n_1}^{n_2} | l_{n_1}^{n_2} \in L_i\} \tag{5.7}$$

$$\textbf{maximize} \quad f_2(x) = \sum_{l_{n_1}^{n_2} \in L_i} -x_{n_1}^{n_2} \tag{5.8}$$

where their order of importance is f_1 and then f_2. In other words, the path computation is considered as an instance of a lexicographic optimization problem. In the next section, first we reformulate our problem as a single-objective Mixed Integer Linear Programming (MILP) one. Then, we propose a novel algorithm based on Branch-and-Cut method to solve it.

5.3 CRP: CamCube Routing Protocol based on Linear Integer Programming

In this section, we describe in depth our optimized path computation algorithm named **CamCube Routing Protocol** (CRP) to solve the problem previously detailed in Problem. 1. Our proposal is based on Branch and Cut algorithm. CRP aims to i) find the optimal path dealing with the bandwidth requirement of flows and ii) maximize load balancing in the CamCube network infrastructure.

To formulate the first objective function as LP problem, we introduce a continuous variable z with the following constraint:

$$z \leq (y_{n_1}^{n_2}) \qquad \forall l_{n_1}^{n_2} \in L_i$$

And the modified objective function as follows:

$$\text{maximize } f_1 = z$$

we then reformulate the multi-objective optimization problem defined in Problem 1 as a single-objective Mixed Integer Linear Programming (MILP) problem:

$$\textbf{maximize } f = Mf_1 + f_2 = Mz - \sum_{l_{n_1}^{n_2} \in L_i} x_{n_1}^{n_2}$$

To retain the order of optimization in the modified problem, we exploit its structure to find a suitable value of the factor M. We note that the values of the minimum residual capacities, after the selection of the path, belong to the finite set $\tilde{\mathcal{Z}}$ defined as:

$$\tilde{\mathcal{Z}} = \{C(l_{n_1}^{n_2}) - B_i \,|\, l_{n_1}^{n_2} \in L_i\}$$

As illustrated in Figure 5.1, we note that M must be chosen such that the slope of the line representing the objective function f is at least equal to that of the line connecting the points $(z_{\text{optimal}}, -|N|)$, $(z_{\text{optimal}} - \Delta z, 0)$ where Δz is the difference between the optimal capacity point and the next-to-optimal capacity one. This is done to ensure that the objective function converges to the optimal point of the original lexicographical optimization protocol f_{optimal}. Besides, we note that $\Delta z \geq \widetilde{\Delta z}$, where

$$\widetilde{\Delta z} = \min\{z_2 - z_1 \,|\, z_2 \geq z_1\,,\, (z_2, z_1) \in \tilde{\mathcal{Z}}\}$$

In other words, M must satisfy the following constraint:

$$M \geq \frac{|N|}{\widetilde{\Delta z}} \geq \frac{|N|}{\Delta z}$$

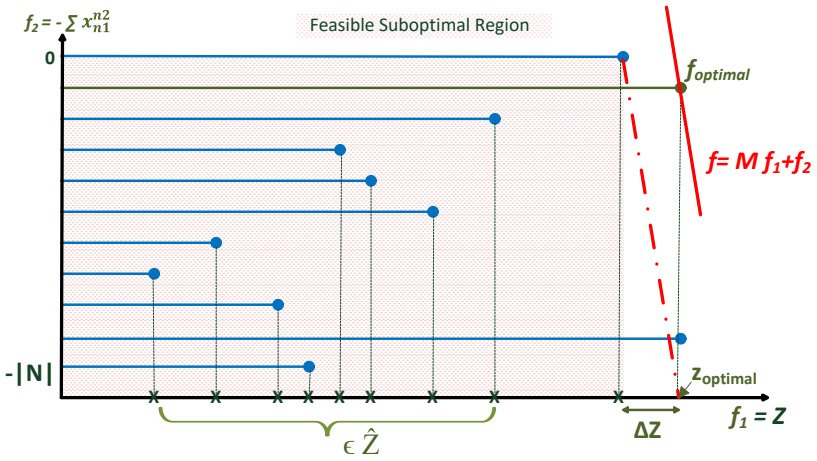

Figure 5.1 – Calculation of the factor M

Problem 2 MILP Reformulation of CamCube Routing problem

$$\textbf{maximize } Mz - \sum_{l_{n_1}^{n_2} \in L_i} x_{n_1}^{n_2}$$

subject to :

$$z \leq y(n_1{}^{n_2}) \qquad \forall l_{n_1}^{n_2} \in L_i$$

$$B_i \cdot x_{n_1}^{n_2} \leq C_i(l_{n_1}^{n_2}) \qquad \forall l_{n_1}^{n_2} \in L(\mathcal{G})$$

$$\sum_{n_1 | l_{n_1}^{n_0} \in L(\mathcal{G})} x_{n_1}^{n_0} \leq 1 \qquad \forall n_0 \in N(\mathcal{G})$$

$$\sum_{n_2 | l_{n_2}^{n_0} \in L(\mathcal{G})} x_{n_0}^{n_2} \leq 1 \qquad \forall n_0 \in N(\mathcal{G})$$

$$\sum_{l_{n_1}^{n_0} \in L(\mathcal{G})} x_{n_1}^{n_0} - \sum_{l_{n_0}^{n_2} \in L(\mathcal{G})} x_{n_0}^{n_2} = \delta_{n_0}^{n_d} - \delta_{n0}^{n_s} \qquad \forall n_0 \in N(\mathcal{G})$$

$$x_{n_1}^{n_2} \in \{0,1\}, z \in \mathbb{R}$$

in order to ensure the convergence of the optimal point. Hereafter, we summarize the reformulation of our problem.

The overall description of our CRP is given in Algorithm 1. We make use of Algorithm 2 which is based on Branch-and-Cut method to solve the constructed MILP model. This algorithm is executed in each flow arrival time in the CamCube network.

Algorithm 1 Pseudo-algorithm of the CamCube Routing Protocol (CRP)

Inputs: $N(\mathcal{G})$, $L(\mathcal{G})$, B_i, C_i, n_s, n_d
Output: path $\mathcal{P}$ for flow $\mathcal{F}_i$
$L_i \leftarrow \{l_{n_1}^{n_2} \,|\, C_i(l_{n_1}^{n_2}) \geq B_i\}$
$\tilde{Z} \leftarrow \{C_i(l_{n_1}^{n_2}) - B_i \,|\, l_{n_1}^{n_2} \in L_i\}$
if $|\tilde{Z}| \leq 1$ **then**
  $\mathcal{P} \leftarrow$ calculated by Shortest Path Routing (Dijkstra)
else
  $\widetilde{\Delta z} \leftarrow \min\{z_2 - z_1 \,|\, z_2 \geq z_1\,, (z_2, z_1) \in \tilde{\mathcal{Z}}\}$
  $M \leftarrow \frac{|N|}{\widetilde{\Delta z}}$
  Construct the MILP as defined in Problem 2.
  Solved Problem 2 as detailed in Algorithm 2.
  $\mathcal{P} \leftarrow \{l_{n_1}^{n_2} \,|\, x_{n_1}^{n_2} \leftarrow 1\}$, where $l_{n_1}^{n_2} \in L_i$
  Installation of path $\mathcal{P}$ via SDN Controller
end if

Algorithm 2 Pseudo-algorithm to Solve the MILP model

Let p^0 the initial problem and $\mathcal{L} = \{p^0\}$ the set of active problem nodes/servers.
Let $x^* = 0$, $\forall\, n_1, n_2 \in N(\mathcal{G})$; $y^* = -\infty$
repeat
  Select and delete a problem p^i from $\mathcal{L}$
  Resolve $\hat{p}^i$ where $\hat{p}^i$ is the LP relaxation of p^i with $\hat{x}_{n_1}^{n_2}$, $\forall n_1, n_2 \in N(\mathcal{G})$ take continuous values between 0 and 1.
  if $\hat{p}^i$ is infeasible **then**
    Go back to step 3
  else
    $\hat{X}$ is the optimal solution with objective value $\hat{y}$
    if $y \leq y^*$ **then**
      Go back to 3
    end if
    if $\hat{x}_{n_1}^{n_2} \in \{0,1\}$,$\forall\, n_1, n_2 \in N(\mathcal{G})$ are integer **then**
      $y^* \leftarrow \hat{y}$
      $X^* \leftarrow \hat{X}$
      Go back to step 3
    end if
  end if
  Search for cutting planes $\mathcal{C}$ violated by x
  if $\mathcal{C} \neq \varnothing$ **then**
    for $c \in \mathcal{C}$ **do**
      $p^i = p^i \cup \{c\}$

23: **end for**
24: Go back to 5
25: **else**
26: Branch to partition the problem into new problems with restricted feasible regions.
27: Add these problems to $\mathcal{L}$
28: Go back to 3
29: **end if**
30: **until** $\mathcal{L} = \varnothing$
31: **return** X^*

5.4 ACO-CRP: CamCube Routing Protocol based on Ant Colony Optimization

In this section, we describe in depth our optimized path computation algorithm named **Ant Colony Optimization for CamCube Routing Protocol** (ACO-CRP) to solve previously detailed in Problem 1. Our proposal is based on the Ant colony optimization meta-heuristic [185][186] to overcome the convergence time issue with CPLEX resolver.

Ant Colony Optimization (ACO) is defined as a meta-heuristic able to solve hard combinatorial optimization problem. The ACO concept is inspired from the real ants behavior when using pheromones to communicate i.e., by following the pheromone trail laying as medium for routing. In fact, similarly to the biological behavior, the ACO is constituted by a colony of agents, named "artificial" ants, and form an indirect communication within this colony through "artificial" pheromone trails. Ants make use of the pheromone trails to find the solution of the problem by calculating the probability expressed in equation (5.9). Thus, the artificial pheromone is defined as a distributed and numerical information that ants adjust all along the algorithm execution in order to emulate their search experience. In addition to the artificial pheromone trails, the artificial ants may use in its probabilistic decisions other heuristic information such as data inputs in order to solve the studied problem. Note that artificial ants build a randomized construction heuristic to make these decisions.

$$P_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \epsilon allowed_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})} \tag{5.9}$$

where α and β two parameters that respectively control the influence of pheromone trails for τ_{xy} and η_{xz}.

In ACO meta-heuristic, artificial ants form the procedures that construct stochastic solution. These procedures build probabilistic and iteratively added solutions. Indeed, in each iteration, these procedures add the resulted solution components to the partial solution by taking into account both the dynamically changed pheromone trails and, if possible, the heuristic information of the problem to be solved. It is noteworthy to say that thanks to the stochastic component within ACO, the ants are able to implement a wide solutions diversity which can help them to

enlarge their solution exploration than greedy heuristics[1]. Also, the use of heuristic information can reinforce the ants exploration and helps them to find the most promising solutions in addition to the use of colony of ants which can increase the algorithm robustness [188]. In order to implement our proposal, we based our algorithm on the Ant colony procedure for combinatorial problems [188] as shown in Algorithm 3. The latter defines a general outline of ACO meta-heuristic application for static combinatorial optimization problems[2]. The Algorithm 3 consists on a main loop based on three main steps for the construction of the solution. After the initialization of ACO parameters, the ants make use of pheromone trails and possibly heuristic information to find the solutions during the first step. Then, the above solutions can be improved during the second phase by providing an optional local solution search. Afterwards, during the third step, the pheromones are updated in order to reveal the ants search experience for the next iteration.
Next, we propose to explain in detail the formulation and the implementation of our ACO-CRP to solve the studied problem:

Algorithm 3 Pseudo-algorithm of skeleton of Ant colony procedure applied to combinatorial optimization

1: Initialization of ACO parameters
2: **while** condition **do**
3: Assemble Ant solutions
4: Establish a local search
5: Update of pheromones for the whole network
6: **end while**

We consider the Graph $\mathcal{G}$ same as described in Section 5.2 where we design the CamCube links $\mathcal{L}$ and nodes compulsory for the resolution of an arrived flow $\mathcal{F}$ between a source node n_s and a destination node n_d. To resolve the aforementioned problem, we define in ACO-CRP algorithm the following metric:

$$\eta = 1 - \frac{B_i}{\hat{C}} \tag{5.10}$$

where B_i is the requested capacity of flow $\mathcal{F}_i$ and $\hat{C}$ is the total capacity of the link. Note that η designs the attractiveness of the ant move from a state x to y [185].
Then, we update the weight of the links of the graph $\mathcal{G}$ according to equation (5.11) where we define τ_{xy} as the trail level of the ant move [185]. In fact, τ impacts the memory of the ant movement for a traveled link where this latter indicator shows how it can impact the decision for the

[1]The greedy algorithms are based on providing a local optimal solution in each stage of the algorithm following problem-solving heuristic [187]. Greedy algorithms may provide a locally optimal solutions that can be near to the global optimal solution in a reasonable convergence time.

[2]The static problem consists on unchanged topology and cost during the algorithm's run time for solving the studied problem. For example, we are running ACO-CRP within an unchanged CamCube topology and where servers keep the same distance and same requested bandwidth (cost) during an execution of the algorithm to find a solution for a considered flow.

previous ant move. The update of τ and its influence for the choice of the next steps are defined by equation (5.12). Note that y_{n1}^{n2} designs the residual bandwidth between nodes $n1$ and $n2$ and B_i represents the requested bandwidth by the accepted flow $\mathcal{F}_i$.

$$\tau = \begin{cases} \frac{y_{n1}^{n2} - B_i}{Nbrhops_{src,y}}, & \text{If } y_{n1}^{n2} > B_i \\ 0 & \text{Otherwise} \end{cases} \tag{5.11}$$

$$\tau_{xy} = (1-\rho)\tau_{xy} + \sum_{k} \Delta\tau_{xy}^{k} \tag{5.12}$$

Our proposal aims to maximize the residual bandwidth while minimizing the number of traveled hops for a treated flow. Indeed, in ACO-CRP, we implement different loops for the search of the near-to-optimal path. As shown in Algorithm 4, in the first loop (see line 5), we make different iterations in order to be closer to the optimal solution. Note that more we increase the number of iterations the closer we are from the optimal solution. However, the increase of iteration number may rise the convergence time. Moreover, in the second loop (see line 7), we aim to maximize the number of possible paths between the source node and the destination node. Finally, for the third loop (see line 12), we aim to avoid moving away from the destination while searching for the next hops. Note that for our experimentations, we fix $\alpha = 2$, $\beta = 1$ [189] and $\rho = 0.5$. In doing so, we aim to allow the searching ant to keep 50% of its memory and 50% of chance while traveling in order to reach the destination.

Algorithm 4 Pseudo-algorithm of the Ant Colony Optimization CamCube Routing Protocol (ACO-CRP)

1: **Inputs:** $N(\mathcal{G})$, $L(\mathcal{G})$, B_i, C_i, n_s, n_d, α, β, ρ, Max_iter, Max_attps, Max_hops,
2: $Max\_Paths\_Per\_iter$
3: **Output:** path $\mathcal{P}$ for flow $\mathcal{F}_i$
4: Calculate η as in (5.10)
5: **for** $iter$ =1 to Max_iter **do**
6: Paths $\leftarrow$ { }
7: **for** $attps$ =1 to Max_attps **do**
8: **if** |Paths| = $Max\_Paths\_Per\_iter$ **then** Go to L1
9: **end if**
10: $x \leftarrow n_s$
11: $Path \leftarrow \{x\}$
12: **for** hop = 1 to Max_hops **do**
13: $Neigh \leftarrow \{ m | L_{xm} \in \mathcal{G}$ and $m \notin Path\}$
14: Calculate P_m $\forall m \in Neigh$ according to equation (5.9)
15: Select $m^* \in Neigh$ randomly with P_{m^*} ▷ We calculate the corresponding P_{m^*} then we select randomly a neighbor m^*.
16: $x \leftarrow m^*$
17: $Path \leftarrow Path + \{x\}$

if $x = n_d$ **then**
  $Paths \leftarrow Paths + \{Path\}$
end if
end for
end for
L1: $\mathcal{P} \leftarrow$ Best Path in Paths ▷ We select according to our criteria: the shortest path among the resulted Paths.
Update τ according to equation (5.12)
end for
Installation of path $\mathcal{P}$ via SDN Controller

5.5 Evaluation of Experimental Results

In this section, we evaluate the performance of our SDN-based proposal CRP and ACO-CRP through extensive experimentations on top of our emulated platform. We discuss the results of our proposals and compare them to those obtained from the traditional link-state shortest-path algorithm (e.g., OSPF) in CamCube and Clos topologies, respectively. It is noteworthy to say that we restrict ourselves to the shortest-path algorithm only, since it is the existing routing protocol considered in literature for the case of CamCube DCN architectures [45]. In addition, the original CamCube [65] architecture makes use of a link-state routing protocol based on the shortest-path routing which motivates the comparison with this approach. Indeed, we evaluate the performance of the aforementioned routing protocols through several performance metrics. The details of the considered evaluation scenarios and performance metrics are presented in Chapter 4.

5.5.1 Performance evaluation for UDP-based traffic

5.5.1.1 QoS analysis

In Figure 5.2, we compare the generated latency of flows $\mathbb{D}$. As we can see, even when the size of CamCube data center increases, our proposals CRP and ACO-CRP outperform the shortest path routing scheme (SPR) and OSPF for respectively CamCube and Clos topologies. In fact, the latency $\mathbb{D}$ for CRP is in between 0.3 and 2 milisecond. Besides, the latency for ACO-CRP records 13.96 ± 0.84 ms to 7.5 ± 0.69 ms for both 100 and 240 servers sized CamCube. Thus, the obtained ACO-CRP latency results are close to the CRP latency values that representing the optimal solutions.
Meanwhile, on one side, the $\mathbb{D}$ values for the SPR vary from 67.37 ± 4.002 ms for 100 servers to 43.965 ± 10.96 ms for 240 servers. On the other side, the latency values of OSPF in Clos topology presents almost 4 times more than $\mathbb{D}$ values for SPR. In fact, the latency records 219.24 ± 28.98 ms for 100 servers and starts by slightly decreasing to reach 159.53 ± 11.81 ms for 240 servers. This can be explained by the fact that servers in CamCube topology is characterised by multipath routing which allow more than one path for a considered server to reach the destination. However, for Clos topology, servers in the same racks are connected to one TOR switch which en-

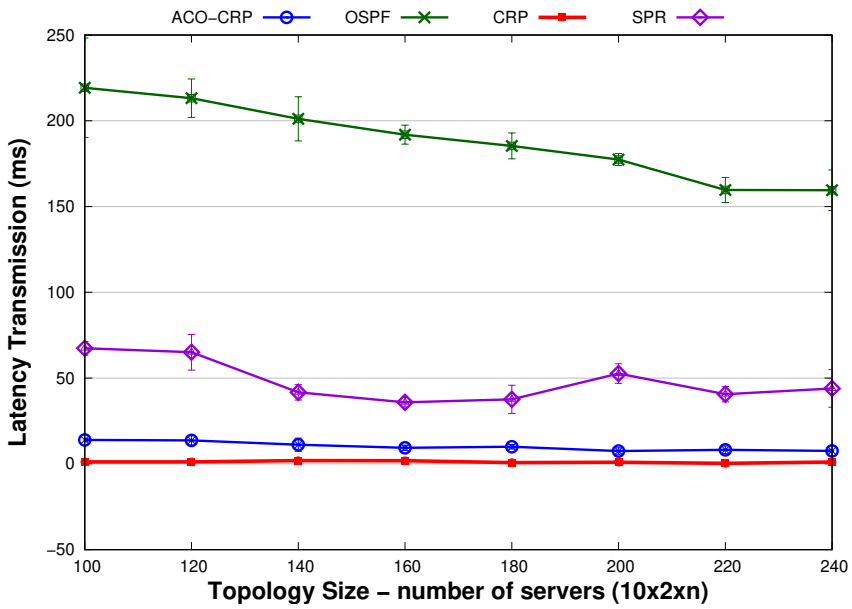

Figure 5.2 – UDP-Latency - $\mathbb{D}$

hances the congestion probability, hence the latency is increased for the transmission of packets. Also, our proposals aim to maximize the residual bandwidth of the network while choosing the shortest path. This trade-off between minimizing the bandwidth utilization and choosing the fast path to the destination can allow CRP and ACO-CRP to minimize the packets latency transmission. Indeed, the obtained results prove that the proposed routing algorithms can decrease by almost 15 times the $\mathbb{D}$ values e.g., when comparing the latency for ACO-CRP and OSPF respectively within 100 servers sized CamCube and Clos DCNs.
Considering the small rising curve behaviour within 200 servers, we can explain it by the fact that we run 5 simulations for each strategy in order to finish the overall testbed in a reasonable time. In this context, we believe that for an increased number of simulations the variation will decrease and follow the predictable behaviour. To dive in the analysis, we will report in Table 5.1 the Relative Error (RE) between from one side the estimated values obtained by ACO-CRP and SPR and on the other side the optimum values obtained by CRP. The complete definition of this latter metric is introduced in Chapter. 4, Section 4.5. In fact, we can notice that for 200 servers sized-toplogy, RE between CRP and SPR noted by RE(CR,SP) is equal to -53.3 whereas the RE between CRP and ACO-CRP is equal to -6.7 which is smaller than the RE(CR,SP) and very close to the optimal solution presented by CRP. Hence, the aforementioned results confirm the curve behaviour illustrated in Figure 5.2.

According to these results, we can conclude that our proposals CRP and ACO-CRP outperform

[1] RE(CR,AC) is the ER ratio between CRP and ACO-CRP latency values
[2] RE(CR,SP) is the ER ratio between CRP and SPR latency values
[3] RE(CR,OS) is the ER ratio between CRP and OSPF latency values

Table 5.1 – Summary of the RE_Delay comparison for unicast flows in CamCube/Clos DCN

Number of servers	RE(CR,AC) [3]	RE(CR,SP) [4]	RE(CR,OS)[5]
200	-6.7	-53.3	-181.9

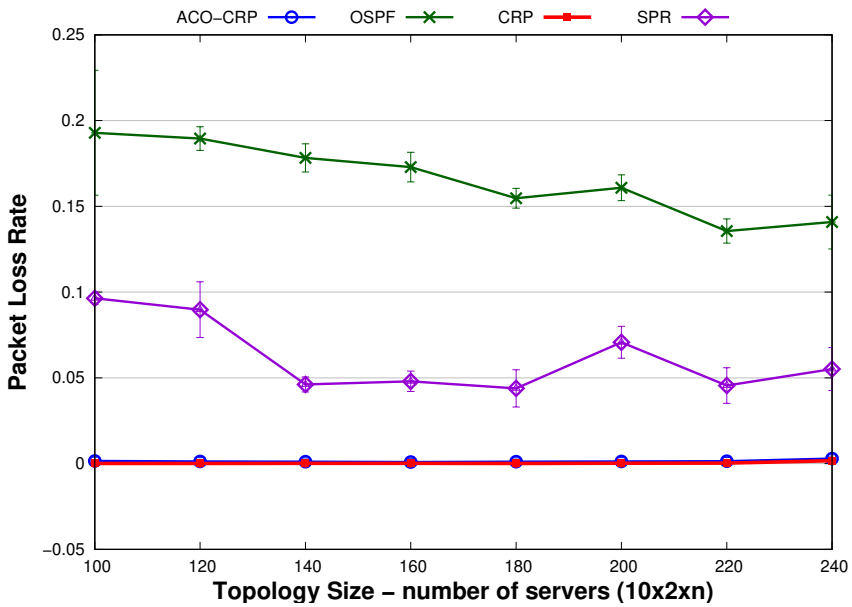

Figure 5.3 – UDP-Packet Loss Rate - $\mathbb{L}$

the SPR routing protocols for both CamCube and Clos DCNs in term of packets transmission latency.

In Figure 5.3, we compare the IP packet loss rate, $\mathbb{L}$. We note a significant gap between our proposals and shortest path routing schemes within CamCube and Clos DCNs.
Thanks to CRP and ACO-CRP, the packets of arrived flows are 100% transmitted as shown in the Figure 5.3 where $\mathbb{L}$ is almost equal to zero within all deployed topologies for both routing protocols. But, for SPR and OSPF, $\mathbb{L}$ is slightly variable. For instance, in CamCube DCN, SPR strategy records 9 ± 0.3 % for 100 servers and 7 ± 0.9 % for 200 servers then decrease to reach 5.5 ± 1.2 % for 240 CamCube sized servers. Besides, for Clos topology, OSPF protocol presents a minimum of packet loss ratio equals to 13.5 ± 0.7 % within 220 servers whereas $\mathbb{L}$ records 19.2 ± 3.6 % within 100 servers.
Moreover, for 200 servers sized topology, we noticed a small variation for SPR showing a small unexpected rise of packet loss rate. Similar to the latency, we believe that this behaviour is caused by the limited number of iterations considered in our simulation's scenarios (due to physical server's capacity limitation). It is also illustrated by the augmentation of confidence interval in comparison with the previous packet loss values for smaller topology sizes. Also, this behaviour is confirmed by the RE_Loss results illustrated in the Table 5.2. In fact, RE(CR,SP) is

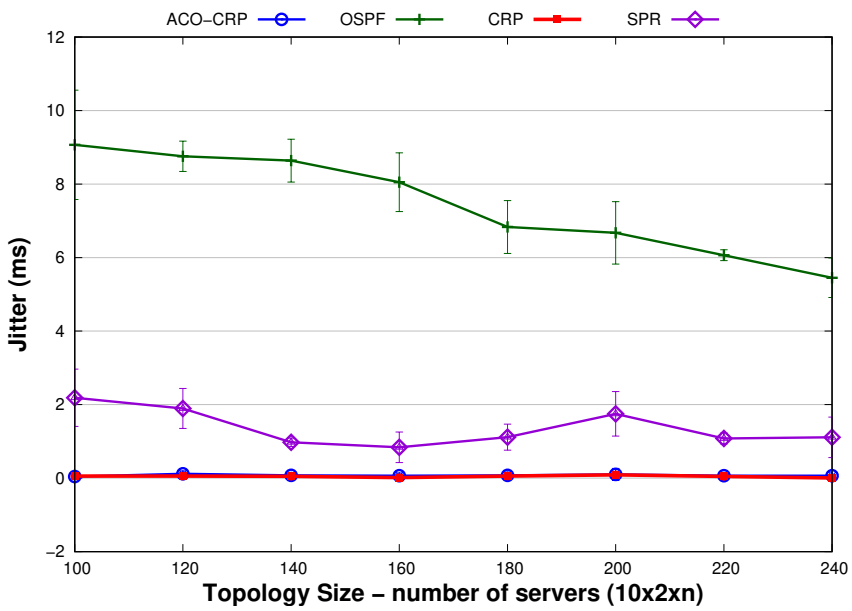

Figure 5.4 – UDP-Jitter - $\mathbb{J}$

equal to -310.8 whereas RE(CR,AC) is tinier with RE_Loss value equal to -4.5 i.e., very close to the optimal results generated by CRP as expected. Moreover, the important gap between OSPF and CRP behaviours, is also indicated in the Table 5.2 where RE(CR,OS) is equal to -707.6 i.e., RE_loss very large compared to RE(CR,AC). The rational explanation behind the aforementioned strategies behaviors is that for an increased CamCube topology size, the congestion is minimized by our proposals as the number of multiple available paths is raised. Hence, the packet loss is tiny. However, for the Shortest Path routing, when the system is stressed by the arrival of multiple flows, the selected paths are rapidly congested since the considered strategy deploys almost the same path to route packets between same nodes of different flows.
The obtained results show that our proposal can better alleviate the congestion, by deeply min-

Table 5.2 – Summary of the RE_Loss comparison for unicast flows in CamCube/Clos DCN

Number of servers	**RE(CR,AC)**	**RE(CR,SP)**	**RE(CR,OS)**
200	-4.5	-310.8	-707.6

imizing the packet loss, within the selected links for routing in comparison with the existing protocol schemes in both CamCube and Clos topologies.

In Figure 5.4, we illustrate the jitter $\mathbb{J}$ for the different studied strategies. It is noteworthy to see the considerable gap between jitter values where OSPF strategy within Clos DCN significantly increases the jitter in comparison with the evaluated protocols within CamCube DCN. For instance, for 100 sized topology,$\mathbb{J}$ is equal to 9.06 ± 1.48 ms for 100 servers and gradually

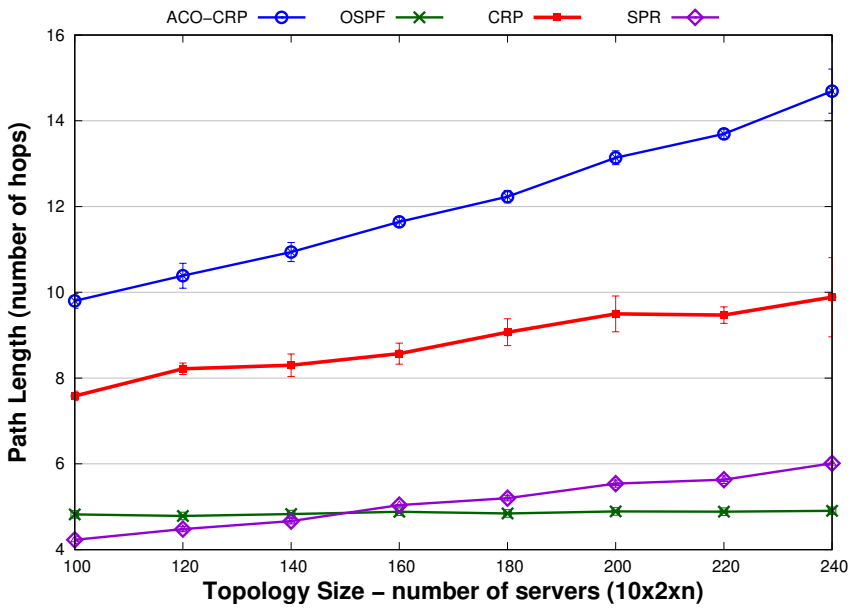

Figure 5.5 – UDP-Path Length - $\mathbb{P}$

decreases to reach 5.45 ± 0.53 ms for 240 servers, whereas, SPR scheme registers jitter values variation equal to 2.18 ± 0.77 ms and 1.109 ± 0.54 ms respectively between 100 and 240 servers sized CamCube. Moreover, it is noteworthy to see that our proposals outperform both SPR and OSPF by significantly decreasing the jitter to almost zero for the overall tested topologies e.g., $\mathbb{J}$ is equal to 0.005 ± 0.0009 ms for CRP and 0.06 ± 0.09 ms for ACO-CRP within 240 CamCube servers for each.
Since the jitter metric represents the variation between the arrival of packets in time to the destination due to routes changes or congested links. Hence, congestion can directly impact the variation of jitter which can explained the obtained results. Indeed, as we deduced from the aforementioned results in terms of latency and packet loss, topologies performing Shortest path routing are suffering from congestion due to the absence of congestion control mechanisms within them. Consequently, according to these results, we can deduce that the more Clos/CamCube links are congested implies more the jitter values are increased. Also, we can confirm that our proposals successfully reach its objectives i.e., by improving the QoS performance of CamCube topology: minimizing the jitter in addition to the latency and the packet loss ratio.

In Figure 5.5, we illustrate the average path length $\mathbb{P}$ with respect to the dimension of CamCube data center. Our proposals generate paths slightly greater then the shortest path which is an expected behavior. In fact, as we mentioned in Chapter 2, CamCube design is characterized by a long routing path diameter. For example, with dimension 240, the average number of hops generated by CRP reaches a maximum value of 9.88 ± 0.92 against 6.01 ± 0.12 for the Shortest Path approach. Also, both analyzed approaches show same overall behavior for all topology sizes.
However, ACO-CRP generates slightly increased paths length in comparison with CRP: maximum

of 5 hops more than the CRP paths length for 240 servers sized CamCube. In fact, for this topology, $\mathbb{P}$ is equal to 14.69 ± 0.51 and 9.88 ± 0.92, respectively. The latter behavior can be explained by the fact that ACO-CRP is based on the Ant colony, so there is a searching phase where the Ants explore more links in order to find the near-to-optimal destination. In fact, the setting of the searching phase related parameters (e.g., iterations and attempts number) should find a trade-off between being close to optimality and minimizing the convergence time to a reasonable value.
Also, it is noteworthy to notice that both ACO-CRP and CRP have similar behaviour where the first protocol achieves our performance goal by presenting values near-to-optimal solution.
Besides, for Clos topology, the path length provided by OSPF record almost constant behavior where $\mathbb{P}$ is equal to 4.82 ± 0.02 for 100 servers and 4.90 ± 0.04 for 240 servers. This can be related to the Clos design. Indeed, servers in the same racks are connected via the same ToR Switches and servers from different racks are interconnected with the same Spine Switches. Hence, the number of travelled hops for inter-rack and intra-rack communications still the same which explain the constant behaviour of the path length of the considered topology and minimized in comparison with strategies deployed in CamCube DCN. Also, considering the SPR, it is noteworthy to see that the path length is gradually increasing while CamCube topology size increases meantimes. In fact, for the same traffic density i.e., number of arrived flows per second, the number of available links increases by the augmentation of the topology sizes whereas the traffic density is keeping the same which can explain the illustrated behavior.
Furthermore, CRP's cost is really negligible in terms of number of hops while the gain is considerable in terms of latency, IP packet loss and jitter. Indeed, considering the requested and residual bandwidth of the flow and the physical link respectively, the judicious optimization resolution improves in depth the quality of service satisfaction. Then, we can conclude that the obtained results are very satisfying.

In the next section, we evaluate the performance of our proposals in comparison with OSPF and SPR by varying the traffic density for 100 servers.

5.5.1.2 Impact of the traffic density

Figure 5.6 illustrates the End to End delay (E2E) which is the total delay from the arrival of the flows until the end of its packet transmission and leaving the system. It is noteworthy to see that CRP significantly increases the total delay by reaching 236.08 ms for a density $\lambda_f = 51$ f/s in comparison with ACO-CRP where E2E is equal to 100.42 for the same density. The rational explanation behind this behavior is that CRP relies on Cplex solver to find the optimal solution. The latter solver is known by its long convergence time in order to find the optimal solution of the considered problem. To overcome the latter issue, ACO-CRP is proposed, and relying on meta-heuristic resolution which minimizes the convergence time as illustrated in Figure 5.6.
Moreover, for SPR and OSPF protocols, the E2E delay is almost similar for the different density values e.g., when $\lambda_f = 12$ f/s, E2E delay is equal to 57.17 ms for SPR and 55.14 ms for OSPF.
Consequently, we can deduce that both SPR and OSPF propose a fast resolution but routes flows packets using the closest servers to the source without checking the links availability and avoiding the bottleneck which is not good in terms of QoS metrics.

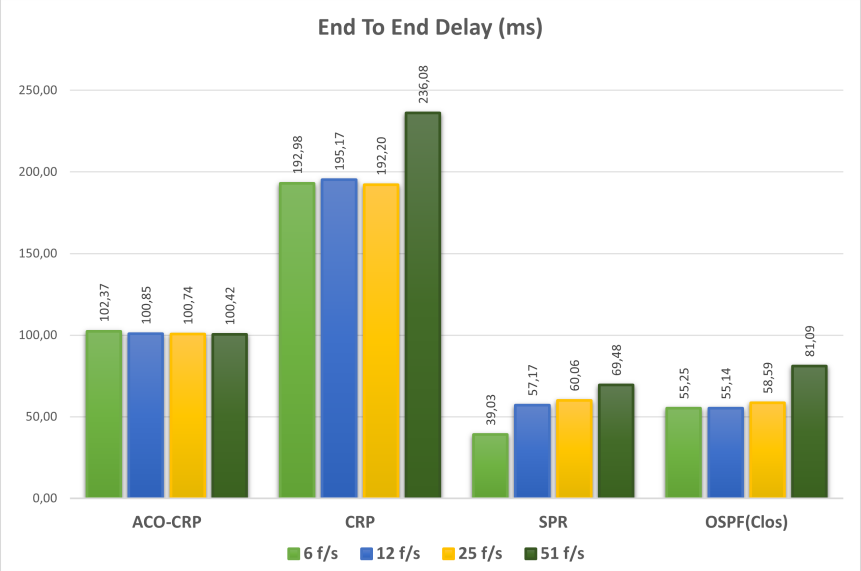

Figure 5.6 – UDP-Average of E2E Delay for different λ_f

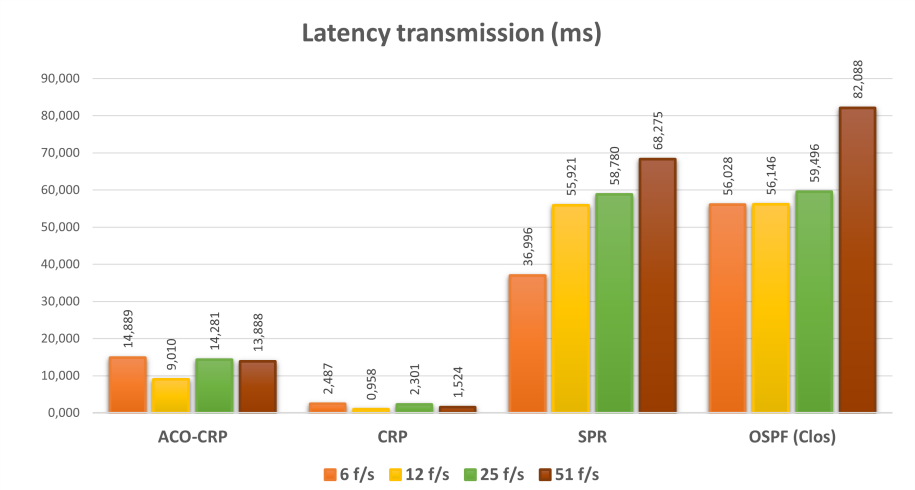

Figure 5.7 – UDP-Qos: Average Latency for different λ_f

Similarly to Figure 5.2, Figure 5.7 shows almost the same behavior of the studied routing protocol schemes in term of latency despite the different values of traffic density. For instance, for ACO-CRP, the latency value varies from 9.010 ms for $\lambda_f = 12$ f/s to 14.28 ms for $\lambda_f = 25$ f/s. Besides, ACO-CRP increases the latency by almost 10 times the latency values for CRP scheme e.g., $\mathbb{L}$ is equal to 0.958 ms for $lambda = 12$ f/s, however the behavior of ACO-CRP still provides near-to-optimal latency values in comparison to the optimal solutions provided by CRP strategy. However, for SPR and OSPF, the latency values are significantly increased e.g., more than 50

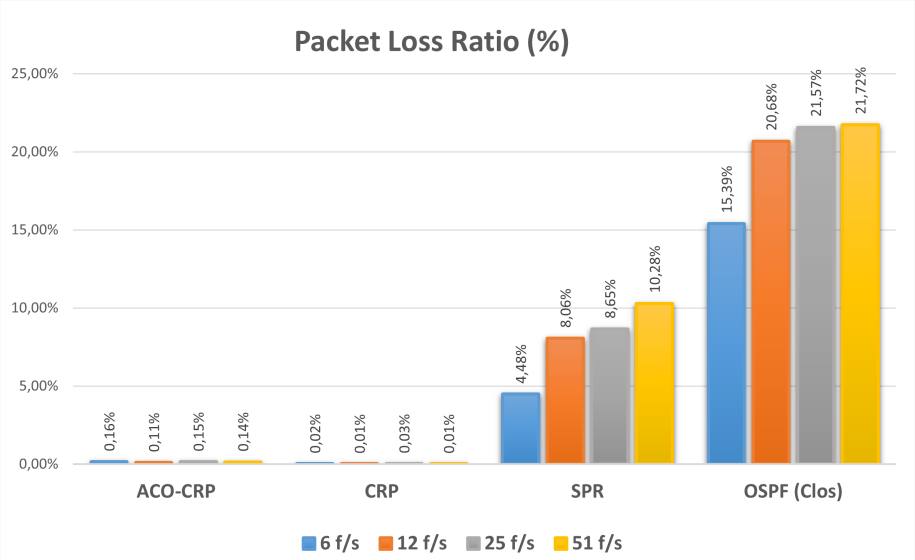

Figure 5.8 – UDP-QoS: Average Packet loss for different λ_f

times for SPR with latency value equals to 68,27 ms for $\lambda_f = 51$ f/s against 1.524 ms for CRP considering the same traffic density. Similarly, OSPF scheme generates more than 80 times increase of the latency values in comparison with CRP i.e., 82.088 ms against 1.524 ms for $\lambda = 51$ f/s, respectively. Also, it is noteworthy to see that when density traffic increases then the latency values also increases for both SPR and OSPF strategies. The latter behaviour can be explained by the fact that the augmentation of the traffic density for a corresponding topology, the CamCube and/or Clos links are congested which impacts the latency values.

Similarly, Figure 5.8 illustrates the packet loss rate which presents the same behavior for different routing protocols while varying the traffic density. In fact, the packet loss rate records its maximum values for the most congested links in both CamCube and Clos topologies when performing respectively SPR and OSPF due to the traffic overload. For instance, when the traffic density increases, the packet loss increases simultaneously e.g., for SPR, the packet loss ratio records 4.4% for $\lambda_f = 6$ f/s and reach 10.28% for $\lambda_f = 51$ f/s. Contrarily, our proposed algorithms i.e., SPR and ACO-CRP deeply decrease the packet loss within CamCube network by reaching less than 1% for overall traffic density values.

In conclusion, our proposals CRP and ACO-CRP can provide a better QoS performance in terms of latency and packet loss independently to the topology size and the number of arrived flows per seconds in the system, whereas, the latter proposals increase the convergence time in comparison to OSPF and SPR within Clos and CamCube DCNs, respectively.

5.5.2 Performance evaluation for TCP-based traffic

In the same context, we deploy the same environment to study the performance of our proposals for TCP flows in terms of throughput while varying both topology size and traffic density of the system.

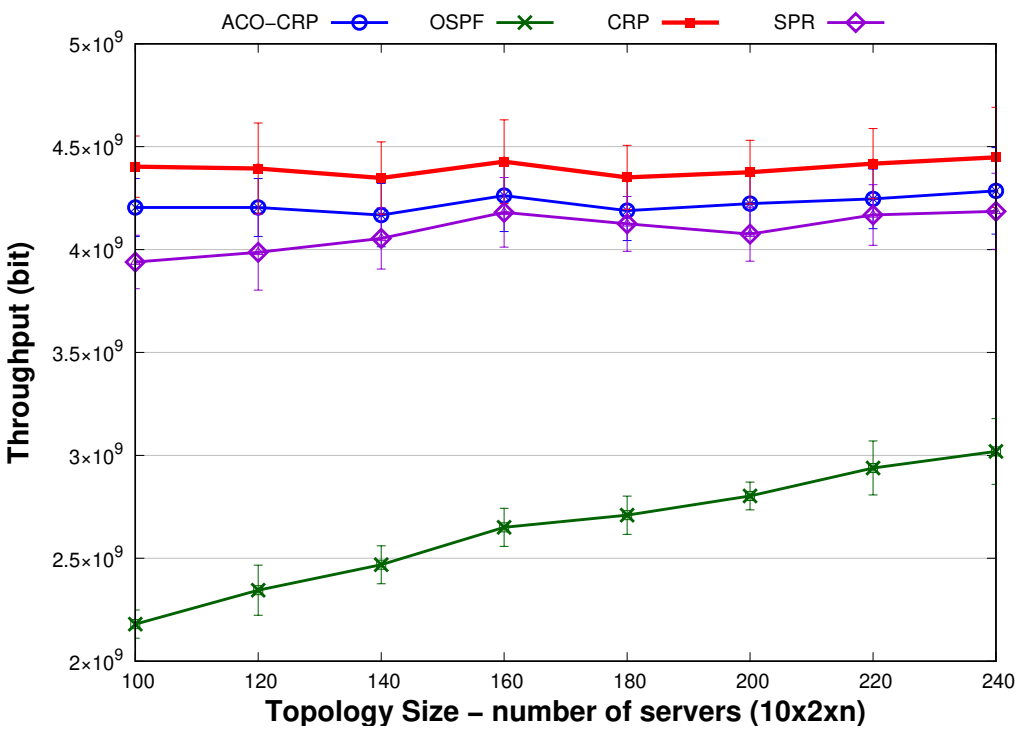

Figure 5.9 – TCP-Throughput-$\mathbb{T}_h$ (bit)

5.5.2.1 QoS analysis

Figure 5.9 illustrates throughput $\mathbb{T}_h$ values of TCP flows performed by the different studied routing protocols. It is noteworthy to see that CRP allows the maximum values of throughput in comparison with the different strategy while proposing the optimal paths in terms of bandwidth. Remember that in our testbed, detailed in Chapter 4, we varied the requested bandwidth between $20-30$ Mbps (uniform distribution) and the duration of each arrived flow is averaged to 180 s. Also, for CRP strategy, the throughput presents an almost same behaviour. For instance, $\mathbb{T}_h$ registers 4.40 ± 0.14 Gbps for 100 servers and reaches 4.47 ± 0.24 Gbps for 240 servers. Moreover, the ACO-CRP and SPR present a close throughput behaviour compared to CRP while ACO-CRP increases the throughput and still close to the optimal solution generated by CRP as expected. Indeed, with 140 servers, ACO-CRP achieves throughput equals to 4.20 ± 0.14 Gbps whereas the $\mathbb{T}_h$ records 4.05 ± 0.15 Gbps. Similarly to the latency and packet loss for UDP flows, the small variation of SPR within 160 and 180 servers can be explained by the limited number of simulations executions where RE_Throughput in Table 5.3 illustrates this variation in comparison to the optimal throughput provided by CRP.
Considering the OSPF scheme within Clos topology, throughput $\mathbb{T}_h$ is significantly decreased compared to CRP protocol. Indeed, OSPF provides a throughput equal to 2.1 ± 0.6 Gbps for a small topology 100 and start progressively increasing until reaching a throughput equal to 3.04 ± 0.15 Gbps near to the optimal throughput within CRP for 240 servers sized topology. This evolution of OSPF throughput is also illustrated in Table 5.3, where throughput is proportional to

Table 5.3 – Summary of the RE_Throughput comparison for unicast Flows in CamCube/Clos DCN

Number of servers	**RE(CR,AC)** [6]	**RE(CR,SP)** [7]	**RE(CR,OS)**[8]
160	+3.72E-02	+5.56E-02	+4.01E-01
180	+3.71E-02	+5.18E-02	+3.77E-01

the topology size i.e., RE(CR,OS) equals to $+4.01E-01$ and $+3.77E-01$ respectively for 160 and 180 Clos/CamCube servers. The rational explanation behind this behaviour is that the topology size increases for a same traffic density. Hence, OSPF exploits more available paths for routing the same traffic load through scalable Clos topology size. Consequently, the throughput is gradually increasing within less-congested links for an expanded Clos DCN.

To conclude, by respecting the requested bandwidth and maximizing the residual capacity of links, our proposals CRP and ACO-CRP provide a better network utilization while ensuring a high load balancing. Consequently, thanks to the latter strategy, the volume of data is progressively increasing and getting close to the CRP performance for a same traffic load density ($\lambda_f = 12$ f/s) and topology size.

5.5.2.2 Impact of traffic density

In this section, we analyse the performance of the studied strategies in terms of throughput by varying the density λ_f from 6 f/s until 51 f/s. To do so, we consider the scenario n°1 illustrated in Table 5.4. Hereafter, we analyse the different obtained results.

Table 5.4 – The different proposed scenario for the testbed

Scenario	**Number of Servers**	λ_f	λ_d	**Requested Bandwidth**	**Max Link Capacity**
Scenario n°1	100	6 f/s	180 s	[20 ..30] Mbps	100 Mbps
Scenario n°1	100	51 f/s	180 s	[20 ..30] Mbps	100 Mbps
Scenario n°2	120	6 f/s	400 s	[40 ..80] Mbps	100 Mbps
Scenario n°2	120	51 f/s	400 s	[40 ..80] Mbps	100 Mbps

Figure 5.10 illustrates the throughput values for the different strategies $\mathbb{T}_a$. It is noteworthy to see that despite the variation of traffic density, the throughput values still the same for each strategy. However, each protocol records a different $\mathbb{T}_a$. For instance, CRP maximizes

[5] RE(CR,AC) is the $\mathbb{RE}$ ratio between CRP and ACO-CRP Throughput values

[6] RE(CR,SP) is the $\mathbb{RE}$ ratio between CRP and SPR Throughput values

[7] RE(CR,OS) is the $\mathbb{RE}$ ratio between CRP and OSPF Throughput values

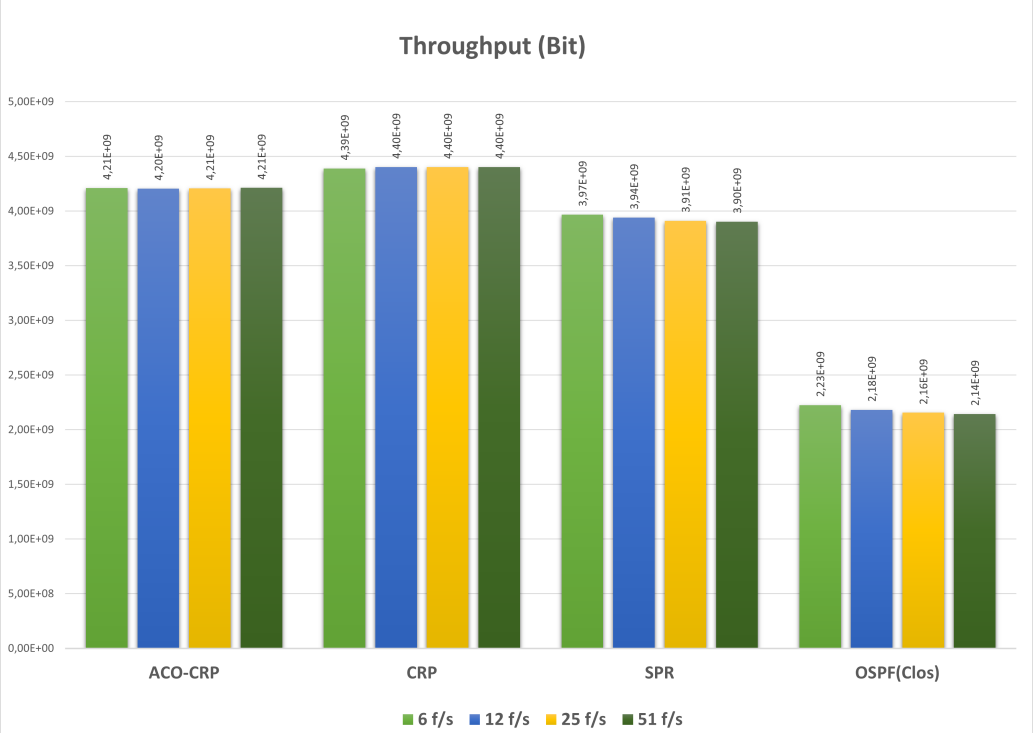

Figure 5.10 – TCP-Throughput(bit) for different λ_f

the throughput value compared to the analyzed routing schemes by reaching 4.39 Gbps and 4.40 Gbps respectively for traffic densities equal to $\lambda_f = 6$ f/s and 51 f/s. For ACO-CRP and SPR, the different protocol schemes are slightly decreasing the throughput for the same traffic load e.g., ACO-CRP presents a throughput equal to 4.21 Gbps whereas SPR records a $\mathbb{T}_h$ equal to 3.9 GBps for $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s, respectively.
However, for Clos topology, the throughput generated by OSPF routing is highly decreased compared to CRP within CamCube. Also, Figure 5.10 shows that by increasing the traffic load, the throughput values decreased progressively. For instance, $\mathbb{T}_h$ is equal to 2.23 Gbps for $\lambda_f = 6$ f/s and decrease to reach 2.14 Gbps for $\lambda_f = 51$ f/s. According to Table 4.3 detailed in Chapter 4 in Section. 4.4.2.3, the number of deployed ToR switches is equal to 8 for 100 Clos servers size. However, for a similar DCN size i.e., 100 servers, CamCube deploys 100 virtual switches performed by servers which can highly impacts the traffic management within the DCN and then explains the doubled values of throughput i.e., from CRP throughput equals to 4.4 Gbps to OSPF throughput equals to 2.18 Gbps with traffic load equal $\lambda_f = 25$ f/s. In conclusion, for a high traffic load within Clos DCN, switches can badly manage the congestion which can deteriorate the throughput respectively. Contrarily, CamCube unicast routing can allow a high throughput by performing the multipath routing. Indeed, the latter allows the routing protocols to exploits more available links and consequently providing a maximized throughput compared to Clos topology.

But, regarding the constant values of $\mathbb{T}_h$ for each strategy, we can make two hypothesis to explain such a behavior i) we make use of a limited number of experimentations (where the number is equal to 5 for a reasonable convergence time purpose) and ii) the deployed testbed scenario cannot sufficient stress the system, so that the flow concurrence is almost limited despite the variation of the number of arrived flows per second. In order to verify the latter hypothesis, we suggest a new testbed scenario in which we increase some inputs values for our system in order to analyze their impact on the protocols behavior.
Thus, as shown in Table 5.4, we define Scenario n°2 as follow: In a first step we increase the number of server of topologies from 100 to 120. Then, we augment the average of requested bandwidth from [20..30] $Mbps$ to [40..80] $Mbps$ i.e., from a minimum equals to the double until a maximum equal to the quadruple compared to Scenario n°1. Last but not least, we increase the duration of the flow transmission from 180 s to 400 s. Finally, we keep the number of experimentation to 5.
According to Scenario n°2, in Table 5.5, results show that the throughput for overall strategies is decreasing while the traffic load is increasing i.e., from $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s. Hence, the CamCube and Clos DCNs are more congested when deploying the new values presented in Scenario n°2. Also, it is noteworthy to see that the overall behavior of the different strategies still the same as in Scenario n°1. In fact, `CRP`, presenting the optimal solution i.e., shows a maximized throughput equals to 23.37 $Gbps$, followed by `ACO-CRP` with $\mathbb{T}_a$ equals to 19.68 $Gbps$. Then, the throughput is deeply minimized within Clos topology where `OSPF` performs only 7.57 $Gbps$. Moreover, it is straight forward to see that the throughput values are highly augmented while comparing the performance of the strategies through the different deployment scenarios i.e., $\mathbb{T}_a$ in Scenario n°2 shows more than 4 times compared to Scenario n°1. Indeed, the latter behavior is impacted by the augmentation of the average of the requested bandwidth for the arrived flows. Hence, we can conclude that by increasing the mean requested bandwidth values by 2.4 (i.e., $60/25 = 2.4$) the throughput of TCP flows can be multiplied by 3 to 4 within CamCube topology i.e., by deploying `CRP`.

Table 5.5 – Summary of TCP Throughput ($\mathbb{T}_h$) in CamCube/Clos DCN

Number of Servers	λ_f	λ_d	**Requested Bandwidth**	**Max Link Capacity**	ACO-CRP **(Gbps)**	**CRP (Gbps)**	**SPR (Gbps)**	**OSPF (Gbps)**
120	6 f/s	400 s	[40 ..80] Mbps	100 Mbps	19.79	23.37	19.68	7.57
120	51 f/s	400 s	[40 ..80] Mbps	100 Mbps	19.76	23.31	19.683	7.54

5.6 Conclusion

In this chapter, we proposed novel SDN-based routing algorithm, named CRP and ACO-CRP, for CamCube-based data centers. The proposed algorithms aim to enhance the network resource allocation inside CamCube topologies and to improve the QoS satisfaction of applications. In the proposed schemes, good performance are achieved by selecting paths ensuring load-balancing in the network infrastructure.
Based on extensive experimentations within our platform, our proposals showed better performance in terms of QoS for both UDP and TCP communications modes by significantly increase the throughput and minimize the packets transmission loss and latency in comparison with the SPR and OSPF respectively within CamCube and Clos topology. But, despite of these prominent QoS performances, our CRP still suffers from a relatively high E2E delay in comparison with the existing studied approaches and ACO-CRP presents a solution that outperforms the basic routing algorithms with acceptable QoS and E2E delay.

Chapter 6
Novel Multicast SDN based Routing Protocols in CamCube DCN

Contents

6.1 Introduction

The majority of the distributed and cooperative applications are simultaneously communicating with each other. Multicast routing provides an efficient way to support Data Center (DC) applications (e.g., replication process of MapReduce jobs, Market data and stock notification apps, IPTV servers, etc.) as it conserves network bandwidth and reduces server load [117]. Also, it can be considered as a solution to avoid the DCN congestion. However, to optimize the exploitation of multicast within the traditional DC networks requires higher performance and capacity from the network devices such as the access capacity of tables and aggregation switches.

In this chapter, we propose and evaluate a novel multicast routing schemes, named M-CRP and ACO-MCRP within CamCube Server-Only DCN architecture while considering the Software Defined Network (SDN) paradigm. We first formulate the multicast routing problem as a lexicographic multi-objective optimization where M-CRP re-formulates the problem as a single-objective ILP. Besides, M-CRP is based on Branch and Cut algorithm and takes into consideration the state of CamCube DCN thanks to the real-time monitoring performed with ONOS controller over the OpenFlow Southbound protocol. Then, we propose ACO-MCRP a new meta-heuristic algorithm based on Ant Colony Optimization. Our objectives are to maximize the residual bandwidth and to minimize the number of relay nodes in the multicast tree. In order to evaluate the performance of our proposals, we emulate the CamCube DCN managed by ONOS SDN controller to optimize the forwarding of multicast.
Finally, based on extensive experimentations, we have run two sets of scenarios. First, we compare M-CRP with the shortest-path approach within the CamCube topology. Second, we have compared CamCube with the traditional DC topology, i.e., Clos. We adopted M-CRP to CamCube and OSPF to Clos. The obtained results from the first set showed that our proposals are better than Shortest Path multicast routing protocols in terms of packet loss, latency, and jitter. The results of the second set showed that CamCube with some intelligent routing algorithm, e.g., M-CRP or ACO-MCRP, is a promising architecture that can support innovative applications in DC such as Big data applications i.e., machine learning jobs for massive data analytic via MapReduce or Spark [190] [191]. Also, we have illustrated that our proposals performed a better multicast tree quality compared to OSPF and SPR.

The remainder of this chapter is organized as follow. First, in Section 6.2, we give the formal description of our multicast routing problem of flows inside CamCube DCN. Then, Section 6.3 details our proposed SDN application Multicast CamCube Routing Protocol (M-CRP). In Section 6.4, we detail our proposal named ACO-MCRP based on Ant Colony optimisation. Afterwards, we provide a full analysis of the obtained results in Section 6.5. Finally, Section 6.7 concludes the chapter.

6.2 Problem Formulation

In this section, we start by giving the definition of our CamCube-based network model. Next, we detail the multicast path computation problem for intra-CamCube DCN traffic flows transmission.

6.2.1 CamCube Network Model

The CamCube servers constitute a directed weighted graph presenting the link capacities in it. We denote this graph as $\mathcal{G} = (N(\mathcal{G}), L(\mathcal{G}))$ where, $N(\mathcal{G})$ is the set of nodes in the graph (i.e., CamCube servers) and $L(\mathcal{G})$ the set of links between two directly linked neighbors. We denote the link from n_1 to n_2 as $l_{n1}^{n2} \in L(\mathcal{G})$. The link l_{n1}^{n2} has an initial bandwidth capacity $\hat{C}(l_{n1}^{n2})$, and we denote its residual bandwidth as $C_i(l_{n1}^{n2})$ at the time instant $\mathcal{T}_i$.

6.2.2 Centralized multicast path computation within CamCube DCN

In this proposal, we consider Constant Bit Rate (CBR) flows. Each flow $\mathcal{F}_i$ requests a fixed bandwidth equals to $B_i = \frac{\mathcal{V}_i}{\mathcal{T}_i}$. The volume of transferred bits is denoted by $\mathcal{V}_i$. It follows a random uniform distribution within duration denoted by $\mathcal{T}_i$ interval which follows the random exponential distribution. We model the arrival rate of $\mathcal{F}_i$ as a Poisson process with density λ_f.

In order to maximize the QoS satisfaction in the network, our objective is to calculate the optimal multicast tree for each $\mathcal{F}_i$ by allocating the requested bandwidth. Then, the SDN controller exploits i) our optimization algorithm (SDN application) proposal and ii) OpenFlow rules to respectively calculate and install the multicast routing tree within CamCube DCN.

The aim of this chapter is to maximize the QoS of flows in CamCube DCN, we propound a multicast routing tree that minimizes the number of links while maximizing the minimal residual bandwidth in DCN. Besides, our proposal maximizes the satisfaction in term of the requested bandwidth and enhances the provider's revenue while respecting the Service Level Agreement.

To be more specific, for flow $\mathcal{F}_i$ from source n_s to the set of destinations $\mathcal{D}$, we search for a multicast tree that maximizes the residual bandwidth with the fewest possible number of links. Then, we consider the binary variable x_{n1}^{n2} equals to 1 when the link l_{n1}^{n2} is selected for the tree path allocation and equals to 0 otherwise. This can be stated formally as the following objective functions.

$$\textbf{maximize} \left[\min \{y_{n1}^{n2} | l_{n1}^{n2} \in L(\mathcal{G})\right]$$

Then

$$\textbf{minimize} \sum_{l_{n1}^{n2} \in L(\mathcal{G})} x_{n1}^{n2} \tag{6.1}$$

We denote y_{n1}^{n2} as an auxiliary variable quantifying the residual capacity of the link, expressed as:

$$y_{n1}^{n2} = C_i(l_{n1}^{n2}) - B_i \cdot x_{n1}^{n2} \tag{6.2}$$

where $C_i(l_{n1}^{n2})$ should be greater than the requested bandwidth B_i and represents the capacity of all links l_{n1}^{n2}. Note that only links l_{n1}^{n2} with sufficient capacity $C_i(l_{n1}^{n2})$ in terms of bandwidth can be considered to build the multicast routing tree in our system. Formally,

$$\forall l_{n1}^{n2} \in L(\mathcal{G}) \qquad B_i \cdot x_{n1}^{n2} \leq C_i(l_{n1}^{n2}) \tag{6.3}$$

The link selection is subjected to the following constraints in order to respect the **multicast tree structure**. First, we can select at most one link entering **into** the node n except at the source n_s where no link is selected as follow:

$$\forall n \in N(\mathcal{G}), \qquad \sum_{l_{n1}^{n2} | n2 = n} x_{n1}^{n2} \leq 1 - \delta_n^{n_s} \tag{6.4}$$

To construct the routing tree, we must have at least one link that goes **out** the source node n_s. Formally,

$$\forall n_s \in N(\mathcal{G}), \qquad \sum_{l_{n1}^{n2} | n1 = n_s} x_{n1}^{n2} \geq 1 \tag{6.5}$$

We require the routing tree to pass by all destinations. Formally, this is expressed as:

$$\forall n \in \mathcal{D}, \qquad \sum_{l_{n1}^{n2} | n1 = n} x_{n1}^{n2} = 1 \tag{6.6}$$

In line with the tree structure, we require that each link in the tree must have at most one parent link except for the links that start at the source node n_s.

$$\forall m, \forall n \in L(\mathcal{G}) \quad | \quad m \neq n_s, \qquad \sum_{l_{n1}^{n2} | n2 = m} x_{n1}^{n2} \geq x_m^n \tag{6.7}$$

Similarly, we require each link in the tree to have at most one child except for links terminating in the destination.

$$\forall m, \forall n \in L(\mathcal{G}) \quad | \quad n \notin \mathcal{D}, \qquad \sum_{l_{n1}^{n2} | n1 = n} x_{n1}^{n2} \geq x_m^n \tag{6.8}$$

Our multicast tree problem in the CamCube data center network is summarized in Problem 3.

In fact, the problem is a lexicographic multi-objective optimization one:

$$\textbf{maximize} \;\; f_1(x) = \min \{ y_{n1}^{n2} \mid l_{n1}^{n2} \in L(\mathcal{G}) \} \tag{6.9}$$

$$\textbf{maximize} \qquad f_2(x) = \sum_{l_{n1}^{n2} \in L(\mathcal{G})} -x_{n1}^{n2} \tag{6.10}$$

where $f_1(x)$ is calculated before $f_2(x)$. Thus, the multicast tree computation is expressed as a lexicographic optimization problem.

In the next section, we express our problem as a single-objective Mixed Integer Linear Programming (MILP). Afterwards, we solve it by proposing a novel algorithm, named `M-CRP`, based on Branch-and-Cut.

Problem 3 Multicast Tree Routing problem in SDN CamCube DCN

maximize $[\min\{y_{n1}^{n2} | l_{n1}^{n2} \in L(\mathcal{G})\}]$

Then

minimize $\sum_{l_{n1}^{n2} \in L(\mathcal{G})} x_{n1}^{n2}$

Subject to :

$y_{n1}^{n2} = C_i(l_{n1}^{n2}) - B_i \cdot x_{n1}^{n2}$

$B_i \cdot x_{n1}^{n2} \leq C_i(l_{n1}^{n2}) \quad \forall l_{n1}^{n2} \in L(\mathcal{G})$

$\sum_{l_{n1}^{n2} | n2 = n} x_{n1}^{n2} \leq 1 - \delta_n^{n_s} \quad \forall n \in N(\mathcal{G})$

$\sum_{l_{n1}^{n2} | n1 = n_s} x_{n1}^{n2} \geq 1 \quad \forall n_s \in N(\mathcal{G})$

$\sum_{l_{n1}^{n2} | n1 = n} x_{n1}^{n2} = 1 \quad \forall n \in \mathcal{D}$

$\sum_{l_{n1}^{n2} | n2 = m} x_{n1}^{n2} \geq x_m^n \quad \forall m, \forall n \in L(\mathcal{G}) \mid m \neq n_s$

$\sum_{l_{n1}^{n2} | n1 = n} x_{n1}^{n2} \geq x_m^n \quad \forall m, \forall n \in L(\mathcal{G}) \mid n \notin \mathcal{D}$

6.3 M-CRP: CamCube Routing Protocol based on Linear Integer Programming

In this section, we detail our multicast tree computation algorithm named **Multicast CamCube Routing Protocol** (M-CRP) aims to not only generate the best multicast tree dealing with the bandwidth requirement of flows but also to maximize the load balancing in the CamCube data center network infrastructure.

To get an ILP formulation, we propose to linearize the first objective function which is related to maximizing the minimum residual link capacity. To this end, we introduce a continuous variable z with the following constraint:

$$z \leq (y_{n1}^{n2}) \qquad \forall l_{n1}^{n2} \in L(\mathcal{G})$$

And then, we redefine the objective function f_1 as follows:

$$\text{maximize } f_1 = z$$

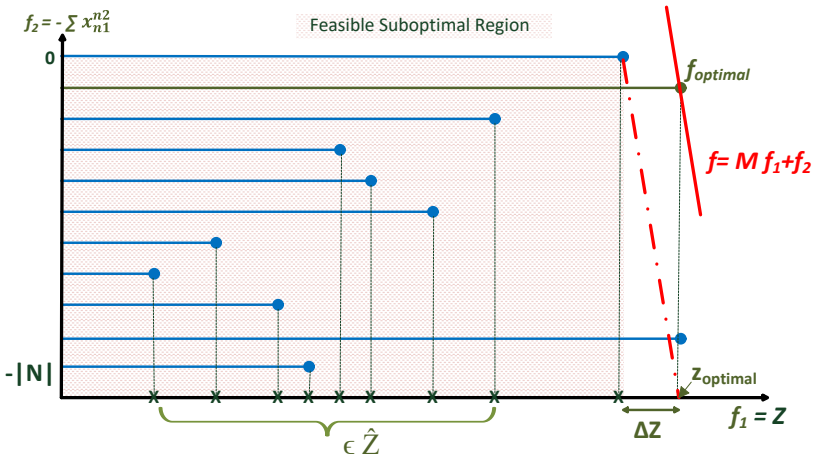

Figure 6.1 – Calculation of the factor M

In the same vein, we propose to reformulate the multi-objective optimization problem defined in Problem 3 as a single-objective Mixed Integer Linear Programming (MILP) as follow:

$$\textbf{maximize} \quad f = Mf_1 + f_2 = Mz - \sum_{l_{n1}^{n2} \epsilon L(\mathcal{G})} x_{n1}^{n2}$$

However, to ensure that the new formulation have the same optimal solution(s) as the previous lexicographical instance, we exploit the structure of the modified problem to set the value of the constant M in order to allow such convergence. We first note that for any feasible solution, (i.e., allocating a route for the flow), the values of the minimum residual capacities are included in the finite set $\tilde{\mathcal{Z}}$ defined as:

$$\tilde{\mathcal{Z}} = \{C(l_{n1}^{n2}) - B_i \mid l_{n1}^{n2} \epsilon L(\mathcal{G})\}$$

Thus, as shown in Figure 6.1, M can be selected such that the (negative) slope of the line representing the total objective function is at least equal to the slope of the line connecting the points $(z_{optimal}, -|N|)$, along with $(z_{optimal} - \Delta z, 0)$ where Δz denotes the difference between the optimal capacity point and the next-to-optimal capacity one.

Then, following the same line of reasoning, we note that $\Delta z \geq \widetilde{\Delta z}$, where $\widetilde{\Delta z}$ is expressed as:

$$\widetilde{\Delta z} = \min\{z_2 - z_1 \mid z_2 \geq z_1 \,, (z_2, z_1) \epsilon \tilde{\mathcal{Z}}\}$$

Stated differently, we must choose M such that it satisfies the following relation:

$$M \geq \frac{|N|}{\widetilde{\Delta z}} \geq \frac{|N|}{\Delta z}$$

Hereafter, we summarize the reformulation of our problem.

Problem 4 MILP Reformulation of CamCube Multicast Routing problem

$$\textbf{maximize } Mz - \sum_{l_{n1}^{n2} \in L(\mathcal{G})} x_{n1}^{n2}$$

subject to :

$$y_{n1}^{n2} = C_i(l_{n1}^{n2}) - B_i \cdot x_{n1}^{n2}$$

$$B_i \cdot x_{n1}^{n2} \leq C_i(l_{n1}^{n2}) \qquad \forall l_{n1}^{n2} \in L(\mathcal{G})$$

$$\sum_{l_{n1}^{n2} | n2 = n} x_{n1}^{n2} \leq 1 - \delta_n^{n_s} \qquad \forall n \in N(\mathcal{G})$$

$$\sum_{l_{n1}^{n2} | n1 = n_s} x_{n1}^{n2} \geq 1 \qquad \forall n_s \in N(\mathcal{G})$$

$$\sum_{l_{n1}^{n2} | n1 = n} x_{n1}^{n2} = 1 \qquad \forall n \in \mathcal{D}$$

$$\sum_{l_{n1}^{n2} | n2 = m} x_{n1}^{n2} \geq x_m^n \qquad \forall m, \forall n \in L(\mathcal{G}) \quad | \quad m \neq n_s$$

$$\sum_{l_{n1}^{n2} | n1 = n} x_{n1}^{n2} \geq x_m^n \qquad \forall m, \forall n \in L(\mathcal{G}) \quad | \quad n \notin \mathcal{D}$$

To resolve the above MILP problem, we propose `M-CRP` scheme described in Algorithm 5. Note that, Algorithm 6 is based on the Branch-and-Cut to resolve the MILP problem. `M-CRP` is an on-line approach and hence executed at every arrival of traffic flow request. Moreover, once Algorithm 6 cannot provide a resolution for an arrived flow request, the latter will be added to a waiting queue in order to be scheduled for a new resolution attempt every Δ_t .

Algorithm 5 Pseudo-algorithm of the Multicast CamCube Routing Protocol (M-CRP)

Inputs: $N(\mathcal{G})$, $L(\mathcal{G})$, B_i, $C_i(l_{n1}^{n2})$
Output: tree $\mathcal{T}$ for flow $\mathcal{F}_i$
$L_i \leftarrow \{l_{n1}^{n2} \mid C_i(l_{n1}^{n2}) \geq B_i\}$
$\tilde{Z} \leftarrow \{C_i(l_{n1}^{n2}) - B_i \mid l_{n1}^{n2} \in L(\mathcal{G})\}$
if $|\tilde{Z}| \leq 1$ **then**
  $\mathcal{T} \leftarrow$ calculated by single source to multi-destination routing
else
  $\widetilde{\Delta z} \leftarrow \min\{z_2 - z_1 \,|\, z_2 \geq z_1\,, (z_2, z_1) \in \tilde{Z}\}$
  $M \leftarrow \frac{|N|}{\widetilde{\Delta z}}$
  Construct the MILP as detailed in Problem 2.
  Solve Problem 2 with Algorithm 2.
  if not solved **then**
    Go back to 11 every Δ_t
  end if
  $\mathcal{T} \leftarrow \{l_{n1}^{n2} | x_{n1}^{n2} \leftarrow 1\}$, where $l_{n1}^{n2} \in L(\mathcal{G})$
  Installation of multicast tree $\mathcal{T}$ via SDN Controller(ONOS)
end if

Algorithm 6 Pseudo-algorithm to Solve the MILP model

Let t^0 the initial problem and $\mathcal{L} = \{t^0\}$ the set of active problem nodes/servers.
Let $x^* = 0$, $\forall l_{n1}^{n2} \in L(\mathcal{G})$; $y^* = -\infty$
repeat
  Select and delete a problem t^k from $\mathcal{L}$
  Resolve $\hat{t}^k$ where $\hat{t}^k$ is the LP relaxation of t^k with $\hat{x}_{n1^{n2}}$, $\forall n1, n2 \in N(\mathcal{G})$ take continuous values between 0 and 1.
  if $\hat{t}^k$ is infeasible **then**
    Go back to step 3
  else
    $\hat{X}$ is the optimal solution with objective value $\hat{y}$
    if $y \leq y^*$ **then**
      Go back to 3
    end if
    if $\hat{x}_{n1}^{n2} \in \{0, 1\}$,$\forall l_{n1}^{n2} \in L(\mathcal{G})$ are all integers **then**
      $y^* \leftarrow \hat{y}$
      $X^* \leftarrow \hat{X}$
      Go back to step 3
    end if
  end if
  Search for cutting planes $\mathcal{C}$ violated by x

20: **if** $\mathcal{C} \neq \varnothing$ **then**
21: **for** $c \in \mathcal{C}$ **do**
22: $t^i = t^k \cup \{c\}$
23: **end for**
24: Go back to 5
25: **else**
26: Branch to partition the problem into new problems with restricted feasible regions.
27: Add these problems to $\mathcal{L}$
28: Go back to 3
29: **end if**
30: **until** $\mathcal{L} = \varnothing$
31: **return** X^*

6.4 ACO-MCRP: CamCube Routing Protocol based on Ant Colony Optimization

Similarly to ACO-CRP detailed in Chapter 5, we get inspired from the Ant Colony Optimization (ACO) i.e., Algorithm 3 in Section 5.4 in order to propose our new Multicast CamCube Protocol (ACO-MCRP) in Algorithm 7 inspired by [192] [193] . We make use of the same equations to calculate the ACO parameters such as η, τ and the Ant probability P^k_{xy}, respectively detailed in Chapter 5, in equations (5.10), (5.11), (5.12) and (5.9).

Our proposal aims to maximize the residual bandwidth in the network while minimizing the number of intermediate nodes in the resulted trees to transport the traffic of flows $\mathcal{F}$ from selected sources n_s to their destinations designed by the set $\mathcal{D}$. Moreover, we keep the same scenario for testing the efficiency of our proposal as for ACO-CRP. Also, for each iteration, we built the resulted tree via the function detailed in Algorithm 8. By the end of each iteration, we updated the values of deposited pheromones i.e., η, τ in order to guide the next arrived ants to find the optimal solution. Then, note that we kept α, β [189] and ρ equal to the same values as described for the ACO-CRP in Chapter 5, i.e., respectively equal to 2, 1 and 0.5.

Algorithm 7 Pseudo-algorithm of the Ant Colony Optimization CamCube Routing Protocol (ACO-MCRP)

1: **Inputs:** $N(\mathcal{G})$, $L(\mathcal{G})$, B_i, $C_i(l^{n2}_{n1})$, n_s, D, α, β, ρ, Max_iter, Max_attps,
2: Max_hops, $Max\_Trees\_Per\_iter$
3: **Output:** Best tree $\mathcal{T}$ for flow $\mathcal{F}_i$
4: Calculate η as in 5.10
5: **for** $iter$ =1 to Max_iter **do**
6: Trees $\leftarrow \{\}$
7: **for** $attps$ =1 to Max_attps **do**
8: **if** |Trees| = $Max\_Trees\_Per\_iter$ **then** Go to 21
9: **end if**
10: $complete\_paths \leftarrow$ GET_MULTICAST_TREE$(\mathcal{G}, n_s, D, \alpha, \beta)$ ▷ detailed in Algo. 8
11: **if** complete_paths **then**

12: $tree \leftarrow resulted\_tree$ ▷ construction of the resulted_tree from the complete_paths
13: $Trees \leftarrow Trees + \{tree\}$
14: **else**
15: Go to 19
16: **end if**
17: **if** not Trees **then** Go to 19
18: **end if**
19: next attps
20: **end for**
21: $\mathcal{T} \leftarrow$ Best tree in Trees ▷ We select the best tree according to our criteria: the tree with minimum number of nodes in Trees
22: Update τ according to equation 5.12
23: **end for**
24: Installation of the tree $\mathcal{T}$ via SDN Controller

In order to generate the multicast tree, we define a new function as detailed in Algorithm 8. The generation of the optimized multicast tree go through different steps and phases.
First, we start by adding the source node n_s to each path for each destination of flow $\mathcal{F}_i$ i.e., we built each branch of the tree and then we update the $path\_lengths$. Then, we make use of a new defined function SELECT_NEXT_NODES to find the next hop from the source. Note that SELECT_NEXT_NODES is a function that selects the next hops according to their ACO probability P^k_{xy} detailed in equation (5.9). Besides, we add the selected next node to each path and consequently we update the $path\_length$. Finally, once we find a destination node, we remove the corresponding node from the list of $destination\_nodes$ and we add it to the list of $reached\_nodes$. We also, update the $paths$ list by removing the path found and we add it to the $complete\_paths$. Later, we update the $path\_lengths$. The aforementioned steps will be repeated until we reach the whole destinations in the set $\mathcal{D}$ of flow $\mathcal{F}_i$.
It is noteworthy to mention that we added some control conditions in order to make sure that we treated all exceptions e.g., assuming that we are working in a non empty graph $\mathcal{G}$ or avoid searching paths for source node n_s included in the $blacklisted\_nodes$. Besides, we select the nearest destination node to the source by choosing the shortest path among the resulted paths to reach each destination e.g., we consider the path that contains the destination node as a neighbour (a next hop) to the source node. Hereafter, we present the pseudo-algorithm of the GET_MULTICAST_TREE for further details of implementation.

Algorithm 8 Pseudo-algorithm of GET_MULTICAST_TREE

1: **function** GET_MULTICAST_TREE($\mathcal{G}, n_s, D, \alpha, \beta$)
2: **Input:** $\mathcal{G}, n_s, D, \alpha, \beta$
3: **Output:** complete_path
4: **Initialization:** reached_destination = {}, blacklisted_nodes ={}
5: $number\_of\_final\_destinations = |D|$
6: $paths \leftarrow \{n_s \in paths \quad | \quad |paths| \leftarrow |D|\}$
7: $success \leftarrow Boolean$
8: complete_paths = {}

```
9:   selected_nodes_from_source ← SELECT_NEXT_NODES(G, n_s, |D|, α, β, paths,
10:                                                 blacklisted_nodes, D)
11:  if |selected_node_from_source| == |D| then
12:    for path_id, node in each(selected_nodes_from_source) do
13:      paths[path_id] ← paths[path_id] + node
14:      path_lengths ← {|p|  |  p ∈ paths}
15:    end for
16:    while true do
17:      if (|G| == 0) or (n_s ∈ blacklisted_nodes) then
18:        success ← False
19:        Go to 46
20:      end if
21:      for path_id, path in reversed(paths) do
22:        if path[−1] ∈ destinations_nodes then
23:          get the index of the destination from the list of destination_nodes
24:          Remove the according node from the list of destination_nodes
25:          Add the node to the list of reached_destinations
26:          Remove the according path from the list of paths and add it to complete_paths
27:          Update path_lengths
28:        end if
29:        if |destination_nodes| == 0 then
30:          Go to 45
31:        end if
32:        path_id ← {np  |  np ← min(paths_lengths)}          ▷ Select the shortest path
33:        next_nodes ← SELECT_NEXT_NODES(G, n_s, number_of_nodes, α, β, paths[path_id],
34:                                       blacklisted_nodes, D)
35:        if |next_nodes| == 1 then
36:          Update paths by adding the next_nodes
37:        else
38:          Update the blacklisted_nodes by adding the next_nodes.
39:        end if
40:      end for
41:    end while
42:  else
43:    success ← False
44:  end if
45:  if success then return complete_paths
46:  else return False
47:  end if
48: end function
```

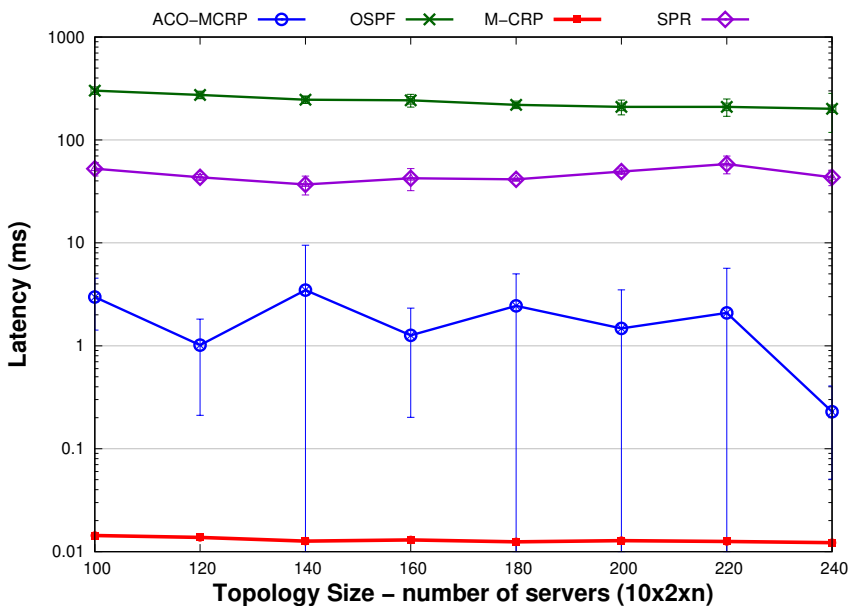

Figure 6.2 – UDP-Latency - $\mathbb{L}$

6.5 Evaluation of Experimental Results

In this section, we start by analyzing figures related to the QoS of our proposals within CamCube in terms of packet loss, latency and jitter compared to the Shortest Path Routing (SPR). Then, we analyse the advantages of CamCube architecture over the traditional Clos topology used in DC (running usually OSPF in the traditional networking scheme without SDN enabled). Then, we will discuss the cost and the quality of multicast trees generated by M-CRP and ACO-MCRP in comparison with multicast trees based on shortest path routing for both CamCube and Clos topologies.

6.5.1 QoS analysis

In Figure 6.2, we show the latency ($\mathbb{L}$) of multicast flows for the different strategies. It is noteworthy to notice that M-CRP deeply minimizes the average of latency transmission within CamCube DCN. For instance, $\mathbb{L}$ is equal to 0.014 ± 0.0009 ms for 100 servers and keep a stable behavior until 240 servers by reaching a latency value equals to 0.012 ± 0.0006 ms.
However, ACO-MCRP increases about 100 times the latency values comparing to CRP but still performs better than SPR and OSPF protocols within CamCube and Clos topologies, respectively. Indeed, we noticed some variation of the latency values for ACO-MCRP curve. As mentioned for the Unicast flows latency, this variation is engendered by the limited number of experimentation in order to achieve a reasonable real execution time. Consequently, ACO-MCRP curve shows the large confidence interval for latency values within 220 servers sized CamCube by registering ± 3.58 ms. Moreover, Table 6.1 summarizes RE_Delay ratio, confirms the latter behaviour

of ACO-MCRP compared to M-CRP by showing a maximized RE(CR,AC) equals to -273.9 for 140 servers.
Besides, to compare the routing protocol based on the Shortest Path for both CamCube and Clos DCNs, we notice that SPR outperforms OSPF where $\mathbb{L}$ equals to 43.38 ± 2.84 ms within 120 CamCube servers for the first strategy and 273.77 ± 19.37 ms for OSPF within the same sized Clos topology. But, SPR still registers high latency values in comparison with the optimal values provided by M-CRP i.e., according to RE ratio table, RE(CR,SP) register a maximum of -4639.5 within 220 CamCube servers.

Table 6.1 – Summary of the RE_Delay comparison for Multicast flows in CamCube/Clos DCN

Number of servers	**RE(CR,AC)**	**RE(CR,SP)**	**RE(CR,OS)**
140	-273.9	-2915.3	-19462.1
180	-195.7	-3334.1	-17625.3
220	-165.1	-4639.5	-16704.5

As we can see, in spite of the increasing number of servers, our proposals outperform the shortest path approaches i.e. SPR and OSPF in both deployed data center networks, by providing a better QoS in term of latency for the generated multicast tree. The latter behaviour can be explained by the fact that M-CRP and ACO-MCRP exploit better the available links when maximizing the residual bandwidth in the network. Also, despite the SPR limitations related to the congestion alleviation, the considered protocol take profit of the multipath routing in CamCube topology to outperform the switch based Clos topology for the selection of the multicast paths.

In Figure 6.3, we illustrate the IP packet loss ($\mathbb{P}$). It is noticeable that the packet loss rate for our proposals M-CRP and ACO-MCRP keeps a constant behavior by significantly minimizing the packet loss where $\mathbb{P}$ is equal to zero for all size topologies (i.e., 100 to 240 servers).
However, for SPR, it is straight forward to see that $\mathbb{P}$ is slowly decreasing even when the size of DCN is increasing (i.e., more resources). For example, when we increase the capacity from 100 to 200 (i.e., 100%), the packet loss slightly decrease and is equal respectively to $7.2 \pm 1.6\%$ and $6.4 \pm 0.5\%$.
Regarding OSPF scheme within Clos topology, we notice that packet loss rate is highly increased by almost 7 times compared with SPR performance within 100 servers (i.e., $\mathbb{P}$ equals to $53.95 \pm 5.8\%$ for OSPF within 100 servers). Then, for an increased Clos DCNs size, $\mathbb{P}$ slowly decreases to get close to SPR approach. By doing so, the gap between SPR and OSPF is continuously minimized e.g., $\mathbb{P}$ is equal to $5.2 \pm 1.3\%$ for SPR and $21.4 \pm 6.3\%$ for OSPF within 240 servers sized topology.
The obtained results show that our proposals M-CRP and ACO-MCRP can better manage the routing of multicast flows by providing more available links capacity which can impact the packet loss rate when transmission. However, since the shortest path strategies, such as SPR and OSPF, cannot take into consideration the network state, same congested link can be chosen for multiple flows, and in an overload scenario, this will yield to a bad performance. In fact, in case

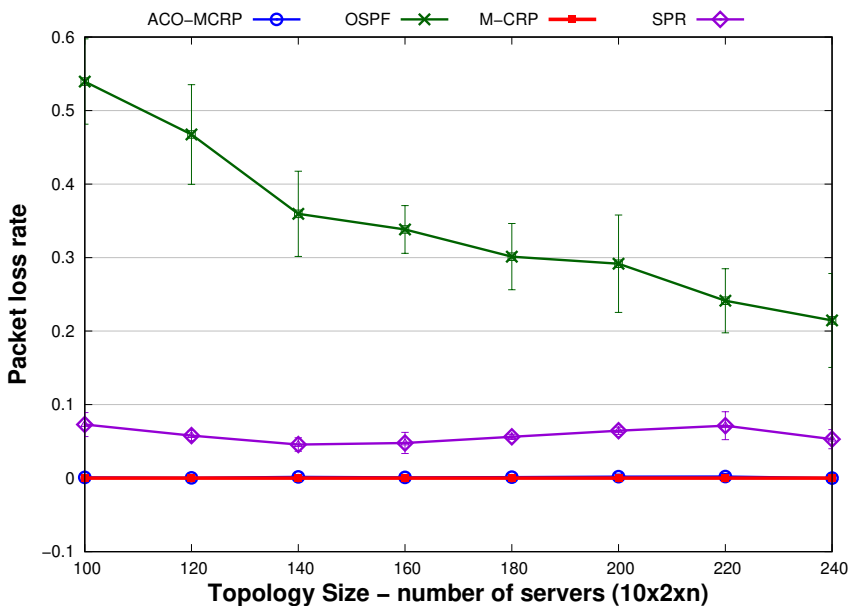

Figure 6.3 – UDP-Packet Loss Rate - $\mathbb{P}$

of traffic overload or a highly solicited nodes for routing, the congestion is increased which by consequence impacts the packet loss ratio. However, for CamCube topology, thanks to multi-path routing presented in this topology, SPR can deeply reduce the packet loss in comparison with OSPF which is performing within Clos DCN. Despite the aforementioned limitation, we can conclude that SPR and OSPF exploits relatively well the available links for an increased data center size which can reduce the packet loss accordingly and get close to the optimal behaviour.

In Figure 6.4, we illustrate the jitter ($\mathbb{J}$) for the different studied strategies. Results show that M-CRP and ACO-MCRP deeply minimize the $\mathbb{J}$ value in CamCube DCN compared to the SPR and OSPF. However, for 120 servers, ACO-MCRP is slightly augmented in comparison with M-CRP by registering a jitter equals to 0.129 ± 0.245 ms for ACO-MCRP and 0.001 ± 0.0001 ms for M-CRP within 120 CamCube servers. Hence, we can conclude that the meta-heuristic proposed algorithm provides accepted results compared to the optimal solution CRP protocol with lower convergence time.
However, it is noteworthy to see that the SPR protocol generates very high jitter values with frequent performance fluctuations, similarly to OSPF when compared to the optimal solution i.e., RE(CR,SP) and RE(CR,OS) equal respectively to -1721.84 and -311.42 for 160 servers as shown in Table 6.2. This performance fluctuations for OSPF and SPR can be explained by the limited number of experimentations elaborated for the generation of these results.
In fact, $\mathbb{J}$ is varying from a minimum equals to 1.5 ± 0.881 ms for 180 servers and reaches a maximum value equals to 2.55 ± 0.427 ms for 160 CamCube servers with SPR. But, the above variation still considered negligible regarding the jitter values for SPR i.e., almost a difference of 1 ms between the minimum and maximum bounds. Furthermore, OSPF performing within Clos

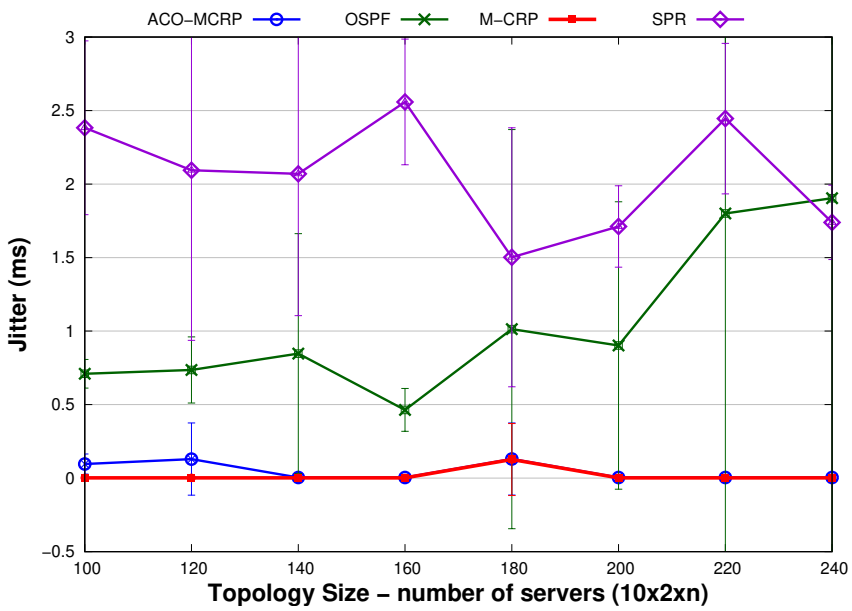

Figure 6.4 – UDP-Jitter - $\mathbb{J}$

topology provides a high jitter compared to our proposals but still lower than $\mathbb{J}$ values obtained with SPR.
The aforementioned behavior can be explained by the fact that SPR relies on shortest path to generate multicast branches which can improve the links utilization and accelerate the bottleneck within small topologies. However SPR is performing within CamCube which is presenting a high number of switches. So, in case of bottleneck, multiple switches are congested and then the packets inter-arrival delay increases. Hence, the jitter is impacted. On the opposite, in our 2-levels Clos topology, the number of solicited switches for routing are limited compared to the CamCube topology but still suffers from congestion, especially for small topologies with limited number of links. Consequently, the jitter values for OSPF are minimized in small topology but increases to reach the SPR values for higher topology.

Table 6.2 – Summary of the RE_Jitter comparison for Multicast flows in CamCube/Clos DCN

Number of servers	**RE(CR,AC)**	**RE(CR,SP)**	**RE(CR,OS)**
160	-1.99	-1721.84	-311.42
220	-1.92	-1691.27	-1244.64

Figure 6.5 illustrates the average number of resolution attempts ($\mathbb{A}$) for all arrived flows performed by our proposal M-CRP compared to the ACO-MCRP, SPR and OSPF. It is straight forward to see that the shortest path strategies for both CamCube DCN and Clos topologies immediately admits all arrived flows even when the network is congested and there is no available resources.

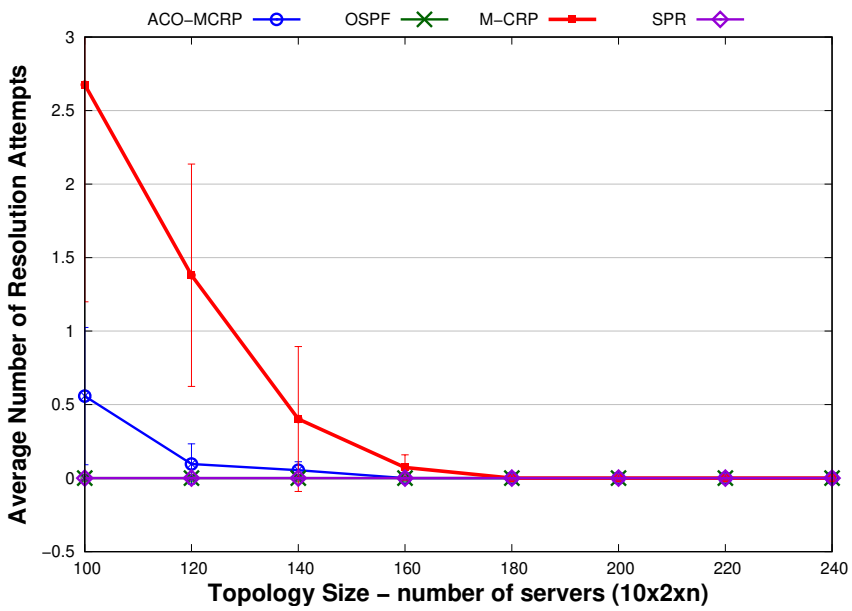

Figure 6.5 – UDP-Average Number of Resolution Attempts - $\mathbb{A}$

Thus, $\mathbb{A}$ is equal to zero for both SRP and OSPF within topology size varying between 100 and 240 servers.
However, our proposals perform an admission control and consider the amount of residual resources in the decision process. For example, for M-CRP and ACO-MCRP, $\mathbb{A}$ is equal to 2.67 ± 1.47 and 0.46 ± 0.46 respectively for DCN size equal to 100 servers. Also, Figure 6.5 shows that M-CRP increases the number of rejected flows compared to ACO-MCRP scheme from 100 to 180 servers within CamCube DCN. The rational explanation behind the latter behavior is that M-CRP relies on Cplex, which is characterized by a long convergence time, to resolve the multicast routing problem. Whereas, ACO-MCRP enhances the convergence time and provides a faster resolution since it relies on meta-heuristics algorithms e.g., Ant Colony Optimization (ACO). By doing so, when MCRP is performing to find the optimal multicast tree for the current flows in the network, the newly arrived flows are rejected. In fact, the convergence time is proportional to the admission system rejects flows. Note that we are running a traffic load density equals to 12 f/s whereas Cplex can exceeds an average of 9 s to build the multicast trees for the treated flows. Thus, the average number of Resolution Attempts is increased contrarily to the fast resolution provided by the ACO-MCRP, i.e., where ACO-MCRP reduces the number of rejected flows. Besides, starting from 180 DCN sized servers, we notice that the average of the resolution attempts for our proposals is decreasing and reaches zero for the rest of CamCube DCN sizes.
Finally, the latter behavior can be explained by the fact that the network is less congested with more resources (i.e., links and virtual switches) and hence M-CRP and ACO-MCRP can easily find the available paths which can consequently decrease their convergence time.

Table 6.3 depicts the average ratio ($\mathbb{T}$) between the number of nodes of the multicast tree

Table 6.3 – Summary of the Multicast Quality tree ($\mathbb{T}$) in CamCube/Clos DCN

Number of Servers	λ_f	λ_d	**Requested Bandwidth**	**Max Link Capacity**	**ACO-MCRP** $\mathbb{T}$	**M-CRP** $\mathbb{T}$	**SPR** $\mathbb{T}$	**OSPF** $\mathbb{T}$
120	6 f/s	400 s	[40 ..80] Mbps	100 Mbps	3.80	3.315	2.68	1.55

and the multicast group. This table indicate that ACO-MCRP shows the maximum ratio in comparison with the different presented strategies by recording 3.315. Hence, we can deduce that ACO-MCRP deploys an average number of nodes almost 4 times more than the multicast group. The latter result can be explained by the fact that ACO-MCRP relies on Ant colony algorithm to discover more nodes in order to search the near-to-optimal solution in comparison with M-CRP based on Cplex resolution. Also, as expected, we notice that the quality tree generated by ACO-MCRP is very close to M-CRP. Besides, regarding SPR and OSPF, results show that both protocols are higly decreasing the ratio T where the number of nodes is minimized within Clos topology while performing OSPF strategy. Indeed, with respect of Clos architecture, T value presented by OSPF can be explained by the fact that the number of traveled nodes is limited for the multicast transmission between a source server and its destinations. In fact, for the intra-rack communications, the transmission of multicast flows is performed locally by the corresponding ToR switch i.e., at most there is 1 switch node travelled to reach the destinations. Also, for inter-rack communications, the source server can communicate at most with ToR and its corresponding Spine switch for the transmission i.e., at most 2 travelled switch nodes. Consequently, the average ratio is equal to 1.55 where the average of number of multicast tree nodes can be equal to the number of multicast group. Moreover, SPR protocol relies on the shortest path algorithm to find the destination which can explain the reduced number of solicited nodes in the multicast tree compared to ACO-MCRP and CRP but still increased in comparison with OSPF. Indeed, remember that SPR is performing within CamCube topology which is known by its long diameter. Thus, the results show that SPR can provide an average of a doubled number of multicast tree nodes compared to the multicast group size. Hence, the ratio T is increased compared to OSPF. Finally, we can deduce that the multicast trees generated by M-CRP and ACO-MCRP are very dense in terms of depth and width compared to the shortest-path-based algorithm e.g., SPR and OSPF. Note that we make use of Scenario n°2, detailed in Table 5.4 in Chapter 5 Section 5.5.2.2, in order to evaluate the quality of the generated multicast tree for the different protocol strategies assuming a stressed system.

6.5.2 Analysis for varied traffic density

In this section, we varied the traffic density i.e., λ_f from 6 f/s to 51 f/s for a size topology equals to 100 servers for both CamCube and Clos topology. Then, we study the impact of this variation on the performance of the different strategies in terms of E2E delay and QoS (latency and packet loss ratio). Hereafter, the different obtained results.

Figure 6.6 shows the overall delay time for the different studied strategies within 100 sized

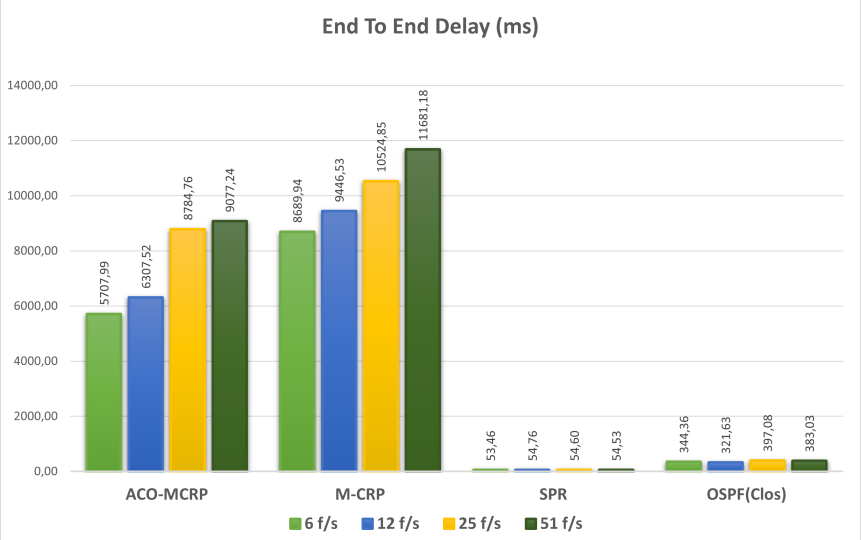

Figure 6.6 – UDP-Average of E2E Delay for different λ_f

topology. It is noteworthy to see that our proposals generate a long E2E time in comparison with shortest path based-protocols within CamCube and Clos DCNs.
Our proposal M-CRP shows a maximized E2E duration which increases following the augmentation of the traffic load with 100 servers sized CamCube. Indeed, E2E time equals to 8.6 s and reach 11.68 s for a number of arrived flows per second varying between $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s, respectively. But, this processing time is considered so long and can be explained by the fact that M-CRP relies on Cplex solver. The latter takes a long resolution time to find the optimal multicast tree for the treated flow. So, despite the high QoS performance of M-CRP, as illustrated in the previous results, our proposal suffers from a long processing time. Consequently, proposing ACO-MCRP based on fast meta-heuristic algorithms can be a solution to overcome this issue.
In fact, as shown in Figure 6.6, ACO-MCRP minimizes the E2E delay time compared to M-CRP as expected. Thus, E2E delay time for ACO-MCRP is equal to 5.7 s for $\lambda_f = 6$ f/s and reach a maximum of 9.07 s for $\lambda_f = 51$ f/s. However, the latter E2E delay time generated by ACO-MCRP still higher than the overall processing time provided by SPR and OSPF.
Indeed, it is straight forward to see that the shortest-path based-routing protocols provide a very short E2E delay compared to our proposals. However, OSPF protocol performed within Clos topology provides an E2E time almost 6 times more than the E2E time of SPR. Actually, the E2E delay time of OSPF is equal to 0.34 s and 0.38 s, whereas the overall processing time is equal to 0.053 s and 0.054 s respectively for a traffic load equal to $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s. First, the latter results show that the E2E delay time is relatively stable within Clos and CamCube DCN for an increased traffic density while performing SPR and OSPF routing schemes. Second, the obtained results show that SPR can deeply minimize the average of overall processing time from the arrival of a considered flow to the system until its departure. Finally, results show that CamCube DCNs outperforms Clos DCNs for routing multicast flows while performing traditional routing

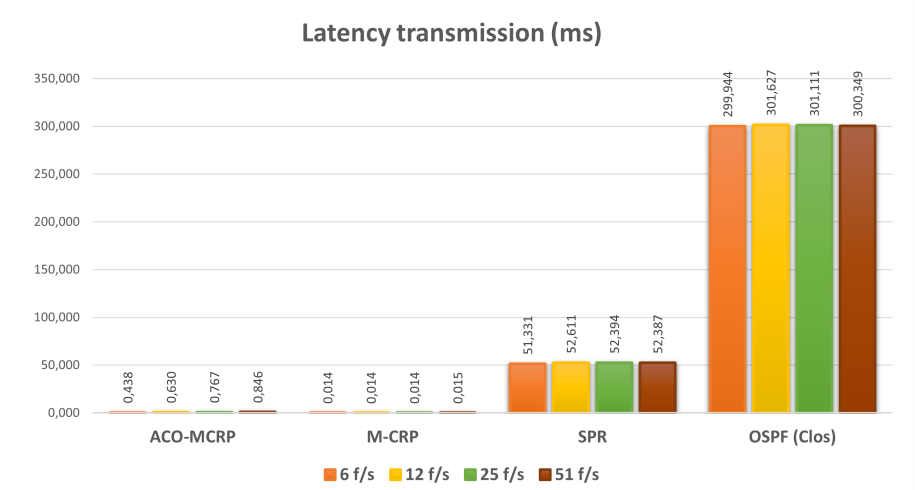

Figure 6.7 – UDP-QoS: Average Latency for different λ_f

schemes e.g., SPR. The rational explanation behind, is that CamCube DCNs provide multipath routing and an increased number of virtualized switches (by forwarding packets through CamCube servers) with high capacity performance which alleviate congestion and increase the number of available links. Hence, the convergence time of protocols based on shortest path routing is minimized within CamCube DCNs.

Table 6.4 – Summary of the Multicast Latency ($\mathbb{L}$) values in CamCube/Clos DCN for Scenario n°2

Number of Servers	λ_f **(f/s)**	λ_d **(s)**	**Requested Bandwidth (Mbps)**	**Max Link Capacity (Mbps)**	**ACO-MCRP $\mathbb{L}$ (ms)**	**M-CRP $\mathbb{L}$ (ms)**	**SPR $\mathbb{L}$ (ms)**	**OSPF $\mathbb{L}$ (ms)**
120	6	400	[40..80]	100	61.76	4.02	148.65	57.21
120	51	400	[40..80]	100	62	5.11	148.65	57.21

Figure 6.7 illustrates the latency of transmitted packets performed by the different strategies through varied traffic density within 100 sized DCNs.
It is noteworthy to see that regarding the traffic load variation, our proposals i.e., M-CRP and ACO-MCRP deeply minimize the latency values as illustrated in Figure 6.2. In fact, M-CRP provides the optimal latency values by providing $\mathbb{L}$ equals to 0.014 ms for $\lambda_f = 6$ f/s and 0.015 ms for $\lambda_f = 51$ f/s. However, our proposal ACO-MCRP increases the latency values up to 30 times compared to the M-CRP latency values. In addition, the meta-heuristic proposal provides a latency value near to the optimal M-CRP one. For instance, for $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s, ACO-MCRP generates latency transmission values equal to 0.43 ms and 0.84 ms. Furthermore, the latter results show that our proposals provide a very performed QoS in terms of latency for running services

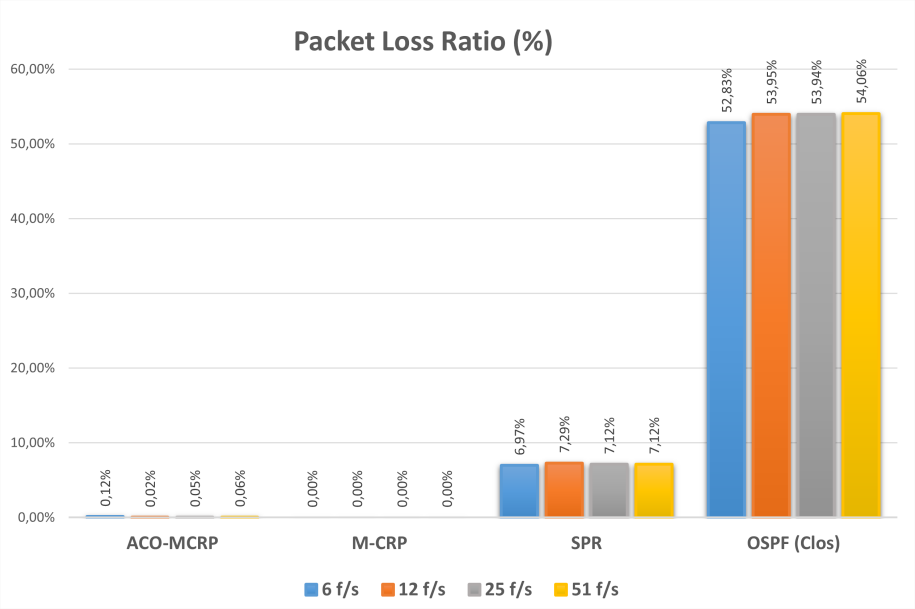

Figure 6.8 – UDP-Qos: Average Loss for different λ_f

compared to the QoS performance generated in real environment e.g., 90% of measured latency of all finished client requests is equal to 140 ms in Google DCNs [194].
However, OSPF protocol significantly increases the latency values by generating up to 5 times compared to SPR routing. Indeed, $\mathbb{L}$ is equal to 299.9 ms and reach 300 ms, whereas, SPR provides latency values equal to 51.3 ms and 52.38 ms respectively for $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s. It is straight forward to notice that the aforementioned strategies provide a stable latency values of transmitted packets despite the augmentation of the traffic load within the same data center size i.e., 100 servers. Then, similarly to the explanation that we provided in Chapter 5 for the constant throughput values, we can explain the behavior of the different strategies shown in Figure 6.7 is caused by a less congested network and a limited number of experimentations. Indeed, as shown in Table 6.4, Scenario 2, the values of the latency is increasing in respect to the augmentation of the traffic density. For instance, the latency is minimized by our proposal M-CRP and increases when λ_f is augmenting i.e., $\mathbb{L}$ records 4.02 ms and 5.11 ms respectively for $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s, whereas, ACO-MCRP registers up to 15 times the latency values for M-CRP. Besides, we can explain that the obtained tiny variation between $\mathbb{L}$ values despite the increased traffic is due to the limited number of experimentations.
To conclude, the performance of multicast routing can be better within CamCube DCN than Clos DCN thanks to the multipath available for routing and the good exploitation of available links within the server-only topology.

In Figure 6.8, we illustrate the average of packet loss ratio for the different aforementioned routing schemes within 100 sized data center infrastructure. First, for M-CRP the packet loss ratio is equal to zero despite the augmentation of traffic density. Second, ACO-MCRP provides a tiny aug-

Table 6.5 – Summary of the Multicast packet loss ($\mathbb{P}$) values in CamCube/Clos DCN

Scenario n°	**Size Topo.**	λ_f (f/s)	λ_d (s)	**Requested Bandwidth** (Mbps)	**Max Link Capacity** (Mbps)	**ACO-MCRP** $\mathbb{P}$ (%)	**M-CRP** $\mathbb{P}$ (%)	**SPR** $\mathbb{P}$ (%)	**OSPF** $\mathbb{P}$ (%)
1	120	6	180	[20..30]	100	0.11	0.04	6.96	46.7
2	120	6	400	[40..80]	100	1.37	0.03	32.6	46.7

mentation of the packet loss rate but still close to the optimal solution provided by the M-CRP. Indeed, for ACO-MCRP, the generated $\mathbb{P}$ is almost stable and equals to 0.05% and 0.06% even for a doubled traffic load i.e., from $\lambda_f = 25$ f/s to $\lambda_f = 51$ f/s.
Last but not least, considering SPR routing scheme within CamCube DCN, the latter routing strategy still generates relatively small packet loss ratio i.e. under 10%. In fact, SPR generates a stable packet loss equal to 7.2% even when we increase the traffic load up to 4 times from $\lambda_f = 12$ f/s to $\lambda_f = 51$ f/s. However, results show that OSPF deeply increases the packet loss ratio within Clos topology i.e., up to 50% for all traffic density. This can be explained by the fact that for the same topology size i.e., 100, switch-based routing such as OSPF in Clos DCN can less manage the bottleneck for an increased number of arrived flows per second. Hence, overloaded switches can engender the network congestion in comparison with CamCube-based routing protocols, especially switches in Clos are less performed and have limited capacity compared to the CamCube servers performing the switch functions. Consequently, the packet loss ratio is maximized for Clos topology, which is confirmed by the obtained results.
Moreover, we test Scenario n°2 as shown in Table 6.5 to analyze the performance of the protocols in a stressed system. Moreover, we can notice the impact of the new inputs values on the approaches behavior. Indeed, it is noteworthy to see that the packet loss rate increases for the overall approaches when we vary λ_f values from 6 f/s to 51 f/s. For instance, $\mathbb{P}$ values for ACO-MCRP registers 10 times greater when we augment the traffic density to $\lambda_f = 51$ f/s i.e., from 0.11% to 1.37%. However, packet loss rate for our proposals still minimized compared to SPR (from 6.96% to 32.6%) and OSPF (from 54% to 46.7%) respectively for $\lambda = 6$ f/s and $\lambda = 51$ f/s.

In conclusion, our proposals can better handle the congestion within CamCube DCNs and enhance the QoS performance compared to the shortest-path based protocols e.g., SPR in CamCube and OSPF in Clos DCN.

6.6 Multicast flows using TCP communication mode

TCP supports only the unicast mode, multicast applications must use the UDP transport protocol [195]. In fact, the multicast cannot be performed via TCP. In fact, multicast is a communication one-to-many and TCP mechanism requires a return of ACK messages to the source from each multicast destination in order to confirm the reception [196, 197].

However, some studies have proposed new transport layer scheme based on TCP supporting multicast flows, such as in [198, 199, 149]. For instance in [199] authors have proposed new

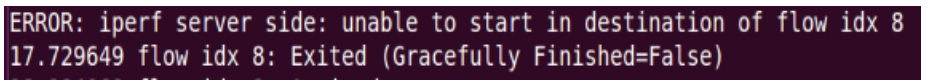

Figure 6.9 – Display of Iperf Error Message

Figure 6.10 – Iperf Message for the destination 4 of Flow 8

transfer protocol scheme based on TCP with a group management in the SDN controller to run multicast traffic efficiently, for only a small group within DCNs.

In our case, and as we make use of iperf tool, we noticed that iperf supports only UDP based traffic [200] for the multicast. Indeed, the official documentation of iperf provides examples and system command for only UDP based multicast flows. In fact, when trying to not force the UDP mode (by omitting -u option) and keep the TCP mode by default, the communication between a given server and its client cannot be established correctly. Figure 6.9 shows the obtained error message for such a flow. For the same experiment, we display the log reported in Figure 6.10. The log shows that the iperf command requires (`-u`) parameter. However, the latter parameter is used only for UDP traffic according to the iperf documentation. For these reasons, we do not perform experiments in this chapter considering TCP as transport layer.

6.7 Conclusion

In this chapter, we addressed the multicast routing problem within SDN based Server only CamCube data center networks and executed our experimentations within a realistic platform made with several open source projects as explained in Chapter 4. Besides, we proposed novel Multicast SDN application named `M-CRP` based on Integer Linear Programming. However, we noticed that `M-CRP` protocol suffers from a high convergence time to generate the optimal multicast trees for the arrived flows. Then, to overcome this issue, we proposed `ACO-MCRP` scheme based on the meta-heuristic Ant Colony Optimization.
The intensive experimental results show that optimization algorithms within SDN based CamCube topology can provide a promising solution for Intra-DC traffic routing issue. Indeed, we have shown that our proposals outperform the traditional multicast shortest path in terms of QoS i.e., latency, jitter, packet loss and the quality of generated multicast tree within CamCube. Note that Clos DCNs increases the convergence time. So, finding a trade-off between convergence time and QoS by judiciously setting the meta-heuristic parameters, is an interesting approach easily operable. Particularly, in case of running experiments based on the applications needs and traffic profile circulating in a given DC.

Chapter 7

Batch-(M)CRP: Novel SDN-Based protocol using Batch scheme in CamCube DCN

Contents

7.1 Introduction

In our previous Chapters 5 and 6, we studied respectively the optimization of unicast and multicast flows in CamCube data center by proposing respectively (CRP and ACO-CRP) and (M-CRP and ACO-MCRP) schemes which are deployed within ONOS SDN controller. The obtained results show that the latter proposals outperform the traditional ones based on shortest path in terms of packet loss, latency and jitter for both CamCube and Clos topologies. However, this online flow treatment needs to request the SDN controller for each arrived flow in the network. So, to manage a voluminous intra-data center traffic and to handle scalability issues, this high frequency of SDN solicitation can lead to performance deterioration. It can also engender the flow loss especially in a congested network. So, the idea of treating arrived flows through **batch routing scheme** can be an alternative solution to overcome these limitations. Indeed, the SDN controller can simultaneously treat a number of flows by calibrating the batch window size. In doing so, we will decrease decision process convergence time of flows while enhancing the system scalability and QoS performance.
In this regard, in this chapter, we start by formulating the batch routing problem as a lexicographic optimization multi-objective one. By this proposal, we aim to maximize the number of accepted flows in our system, to maximize the residual bandwidth, and to minimize the traveled hops to reach the destinations. Afterwards, we propose a new multi-phase batch routing algorithm in CamCube topology denoted by batch-(M)CRP based on SDN application. batch-(M)CRP performs both unicast and multicast flows for CamCube Routing Protocol. Then, in order to simplify the notation we consider i) batch-CRP that treats UDP unicast flows and ii) batch-MCRP performing UDP Multicast flows. We reformulate the problem as a single objective problem in aim to make use of branch and cut algorithm to solve it. batch-(M)CRP retrieves the real-time network state of CamCube DCN using ONOS controller via its Openflow southbound interface. Then, we emulate the CamCube topology managed by ONOS controller and study the performance of our proposal through extensive experimentations by comparing it with the shortest path. The obtained results show that batch-(M)CRP achieves good performances in terms of packet loss, latency and jitter.
The remainder of this chapter is organized as follow. First, in Section 7.2, we describe the batch routing problem of flows within CamCube DCN. Next, Section 7.3 details our proposed SDN application batch CamCube Routing Protocol (batch-(M)CRP). Afterwards, Section 7.4, discusses the experimental obtained results. Finally, Section 7.5 concludes the chapter.

7.2 Problem Formulation

In this section, we propose CamCube-based network model. Afterwards, we explain the batch routing problem within CamCube DCN.

7.2.1 CamCube Network Model

The servers in the DCN form a directed graph where capacities represent the weight of its links. We define the graph as $\mathcal{G} = (N(\mathcal{G}), L(\mathcal{G}))$ where, $N(\mathcal{G})$ designs the CamCube servers (i.e., the

nodes of graph) and $L(\mathcal{G})$ the set of links connecting every two neighbors servers. We design $l_{n1}^{n2} \in L(\mathcal{G})$ as the link directed from $n1$ to $n2$. The initial link l_{n1}^{n2} weight is equal to the initial bandwidth capacity $\hat{C}(l_{n1}^{n2})$. Then, for a time instant $\mathcal{T}_i$, we define its residual bandwidth as $C_i(l_{n1}^{n2})$.

7.2.2 Centralized batch routing within CamCube DCN

In the proposed centralized batch routing, we consider that each flow $\mathcal{F}_i$ is a Constant Bit Rate (CBR) flow and requests a fixed bandwidth defined as $\mathcal{B}_i = \frac{\mathcal{V}_i}{\mathcal{T}_i}$, where $\mathcal{V}_i$ defines the volume of transferred bits following a random uniform distribution within duration $\mathcal{T}_i$. The latter is expressed following the random exponential distribution. Besides, the arrival rate of $\mathcal{F}_i$ is designed as a Poisson process with density defined as λ_f.

The proposed routing algorithm aims to maximize the QoS satisfaction of the network for both unicast and multicast traffic and then the user experience by allocating the needed performances (i.e., requested bandwidth, maximizing the treatment of arrived flows, etc.). In fact, the objective of this proposal is to consider the accepted set of $\mathcal{F}_i$ and to calculate the optimal path/tree to reach the destination by considering the requested bandwidth. The SDN controller exploits the optimization tools and openflow rules to install the calculated path/tree for each flow. Our proposed algorithm aims to maximize the flows acceptance rate in **the batch window** Δ_t by balancing the link bandwidth usage in the infrastructure. Specifically, for each batch window, the arrived flows $\mathcal{F}_i$ from source n_s to the destination n_d, we propound to build a batch path/tree where maximizing the number of treated flows.

Once the flow is admitted, the SDN controller uses first our proposed optimization algorithm and then calculates and install the batch path/tree routing in the DCN.

Besides, we denote $\mathcal{F}$ as the set of arrived flows in the batch window Δ_t to be treated. We consider to define 0-1 variable $x_{n1n2}^{\mathcal{F}_i}$ representing the combined decision related to the selected link and for an accepted flow $\mathcal{F}_i$ from $n1$ to $n2$. Moreover, $\mathcal{A}^{\mathcal{F}_i}$ designs the state of flow $\mathcal{F}_i$, (i.e., if accepted or not). The obtained results are represented by directed path/tree starting by the source node to the destination(s). Thus, we consider the following objective function as:

$$\textbf{maximize} \sum_{\mathcal{F}_i \in \mathcal{F}} \mathcal{A}^{\mathcal{F}_i}$$

Then

$$\textbf{maximize} \left[\min \{y_{n1}^{n2} \mid l_{n1}^{n2} \in L(\mathcal{G})\}\right]$$

Then

$$\textbf{minimize} \sum_{l_{n1}^{n2}} x_{n1}^{n2} \tag{7.1}$$

We maximize the acceptance of the arrived flows in the batch windows by maximizing $\mathcal{A}^{\mathcal{F}_i}$ variable. Moreover, we define y_{n1}^{n2} as an auxiliary variable expressing the residual bandwidth of each link in the CamCube DCN as follow:

$$y_{n1}^{n2} = C_i(l_{n1}^{n2}) - \mathcal{B}_i \cdot x_{n1}^{n2} \tag{7.2}$$

where the capacity of all links l_{n1}^{n2} is expressed by $C_i(l_{n1}^{n2})$. The latter must be greater than the bandwidth requests (i.e., $\mathcal{B}_i$). So, we consider that only links with sufficient capacities for the majority of accepted flows can be selected to build the final solution. Formally,

$$C_i(l_{n1}^{n2}) \geq \mathcal{B}_i \cdot x_{n1}^{n2} \qquad \forall l_{n1}^{n2} \in L(\mathcal{G}) \tag{7.3}$$

Moreover, we must ensure that the set of links selected as path/tree for flow $\mathcal{F}_i$ conforms to the mathematical structure oriented path/tree:

$$\Pi^{\mathcal{F}_i} = \{l_{n1}^{n2} \mid x_{n1n2}^{\mathcal{F}_i} = 1\} \tag{7.4}$$

First, we must ensure that at most one link can **into** nodes except for source node where there is no incoming link. Then we can express this constraints as the following inequality:

$$\sum_{l_{n1}^{n2} | n2 = n} x_{n1n2}^{\mathcal{F}_i} \leq 1 - (\delta_n^{n_s})^{\mathcal{F}_i} \qquad \forall n \in N(\mathcal{G}), \forall \mathcal{F}_i \in \mathcal{F} \tag{7.5}$$

where we used the Kronecker delta function δ defined as: $\delta_x^y = \textsc{IsEqual}(x, y)$ to shorten the notation for the particular case at source $''n_s''$.
The auxiliary variable $\mathcal{A}^{\mathcal{F}_i}$ is linked to the binary variable $x_{ij}^{\mathcal{F}_i}$ by the following constraints which guarantee the constructed path/tree $\Pi^{\mathcal{F}_i}$. The latter is empty only when the flow is not accepted (i.e., $\mathcal{A}^{\mathcal{F}_i} = 0$):

$$\mathcal{A}^{\mathcal{F}_i} \leq \sum_{l_{n1}^{n2} | n1 = n_s} x_{n1n2}^{\mathcal{F}_i} \leq M \cdot \mathcal{A}^{\mathcal{F}_i} \quad \forall l_{n1}^{n2} \in L(\mathcal{G}), \quad \forall \mathcal{F}_i \in \mathcal{F} \tag{7.6}$$

where M is a big constant that satisfies the condition: $M \geq |\{l_{n1}^{n2} \mid n1 = n_s^{\mathcal{F}_i}\}|$
In order to verify the connectedness of $\prod^{\mathcal{F}_i}$, a link l_m^n is selected only when it has a parent link except at the source level. Formally, this is expressed as:

$$\sum_{l_{n1}^{n2} | n2 = m} x_{n1n2}^{\mathcal{F}_i} \geq x_{mn}^{\mathcal{F}_i} \quad \forall l_m^n \in L(\mathcal{G}) \quad | \quad m \neq n_s \tag{7.7}$$

Similarly, link l_m^n should have at least one successor link (i.e., a child) except at destination nodes.

$$\sum_{l_{n1}^{n2} \mid n_1 = n} x_{n1n2}^{\mathcal{F}_i} \geq x_{mn}^{\mathcal{F}_i} \quad \forall l_m^n \in L(\mathcal{G}) \quad | \quad n \notin \mathcal{D}^{\mathcal{F}_i} \tag{7.8}$$

where $\mathcal{D}$ designs the set of destination nodes.
For an accepted flow, we must ensure that path/tree Π^k reaches the destination node of this flow. This restriction is expressed by the following constraint:

$$\sum_{l_{n1}^{n2} \mid n1=n} x_{n1n2}^{\mathcal{F}_i} = \mathcal{A}^{\mathcal{F}_i} \quad \forall \mathcal{F}_i \in \mathcal{F} \quad | \quad n \in \mathcal{D}^{\mathcal{F}_i} \tag{7.9}$$

A link is considered for routing only when it can satisfy the demands of the respective flows as detailed in the following expression:

$$C_i(l_{n1}^{n2})^{\mathcal{F}_i} \geq \sum_{\mathcal{F}_i \in \mathcal{F}} \mathcal{B}_i^{\mathcal{F}_i} \cdot x_{n1n2}^{\mathcal{F}_i} \qquad \forall l_{n1}^{n2} \in L(\mathcal{G}) \tag{7.10}$$

Summarily, our batch routing problem is formulated in Problem 5. Then , we remind that it is a lexicographic multi-objective problem formulated as:

$$\textbf{maximize} \qquad f_1(x) = \sum_{\mathcal{F}_i \in \mathcal{F}} \mathcal{A}^{\mathcal{F}_i} \tag{7.11}$$

$$\textbf{maximize} \quad f_2(x) = \min \{y_{n1}^{n2} \mid \forall l_{n1}^{n2} \in L(\mathcal{G})\} \tag{7.12}$$

$$\textbf{maximize} \quad f_3(x) = \sum_{l_{n1}^{n2}} -x_{n1}^{n2} \mid \forall l_{n1}^{n2} \in L(\mathcal{G}) \tag{7.13}$$

where in equation (7.11), we maximize the number of accepted flow for batch window. Then, in equation (7.12), f_2 maximizes the minimum of residual bandwidth. Finally, f_3 in equation (7.13) minimizes the number of hops involved in the routing. Hence, the routing path/tree is formulated as a lexicographic optimization problem.

In the next section, we present our problem as Mixed Integer Linear Programming (MILP) problem. Later, we propose a multi-phase algorithm named **batch (Multicast) for CamCube Routing Protocol** (`batch-(M)CRP`) based on Branch and Cut algorithm to solve it.

Problem 5 Batch routing problem within CamCube DCN

maximize $\sum_{\mathcal{F}_i \in \mathcal{F}} \mathcal{A}^{\mathcal{F}_i}$

Then

maximize $[\min \{y_{n1}^{n2} \mid l_{n1}^{n2} \in L(\mathcal{G})\}]$

Then

minimize $\sum_{l_{n1}^{n2}} -x_{n1}^{n2}$

Subject to :

$$\sum_{l_{n1}^{n2} | n2=n} x_{n1n2}^{\mathcal{F}_i} \leq 1 - (\delta_n^{n_s})^{\mathcal{F}_i} \qquad \forall n \in N(\mathcal{G}), \forall \mathcal{F}_i \in \mathcal{F}$$

$$\mathcal{A}^{\mathcal{F}_i} \leq \sum_{l_{n1}^{n2} | n1=n_s} x_{n1n2}^{\mathcal{F}_i} \leq M \cdot \mathcal{A}^{\mathcal{F}_i} \qquad \forall l_{n1}^{n2} \in L(\mathcal{G}), \forall \mathcal{F}_i \in \mathcal{F}$$

$$\sum_{l_{n1}^{n2} | n2=m} x_{n1n2}^{\mathcal{F}_i} \geq x_{mn}^{\mathcal{F}_i} \qquad \forall l_m^n \in L(\mathcal{G}) \mid m \neq n_s$$

$$\sum_{l_{n1}^{n2} \mid n1=n} x_{n1n2}^{\mathcal{F}_i} \geq x_{mn}^{\mathcal{F}_i} \qquad \forall l_m^n \in L(\mathcal{G}) \quad | \quad n \notin \mathcal{D}^{\mathcal{F}_i}$$

$$\sum_{l_{n1}^{n2} \mid n1=n} x_{n1n2}^{\mathcal{F}_i} = \mathcal{A}^{\mathcal{F}_i} \qquad \forall \mathcal{F}_i \in \mathcal{F} \mid n \in \mathcal{D}^{\mathcal{F}_i}$$

$$C_i (l_{n1}^{n2})^{\mathcal{F}_i} \geq \sum_{\mathcal{F}_i \in \mathcal{F}} \mathcal{B}_i^{\mathcal{F}_i} \cdot x_{n1n2}^{\mathcal{F}_i} \qquad \forall l_{n1}^{n2} \in L(\mathcal{G})$$

7.3 batch-(M)CRP: Batch scheme in CamCube Routing Protocol

In this section, we propose our algorithm batch-(M)CRP which seeks to maximize the number of flows served (i.e., as expressed in f_1). The algorithm aims to both minimize the residual bandwidth and the number of traveled hops respectively by f_2 and f_3. These functions act as surrogates for enhancing the load balancing and minimizing the delay in CamCube topology.

Before introducing our proposal, we suggest to linearize the second objective function detailed in equation (7.12). To do so, we used an additional auxiliary and continuous variable z to express the new objective function as:

$$\textbf{maximize } f_2' = z \tag{7.14}$$

with the following constraint:

$$z \leq y_{n1}^{n2} \qquad \forall l_{n1}^{n2} \in L(\mathcal{G}) \tag{7.15}$$

Our proposal, to address Problem 5, consists on solving a sequence of three single objective problems. In the first phase, we solve the following problem expressed by equation (7.11) in order to get the optimal value $f_1^{(opt)}$. Besides, in the second phase, we consider f_2' as in equation (7.14) such that $f_2' \leq f_2'^{(opt)}$. The latter gives us the optimal solution with objective value $f_2'^{(opt)}$. Later, in the third phase, we solve the problem detailed in equation (7.13) such that $f_1 \leq f_1^{(opt)}$ and $f_2' \leq f_2'^{(opt)}$.

Note that the problem in each phase is solved by the successful Branch and Cut algorithm [201]. The pseudo-code of the algorithm `batch-(M)CRP` is formulated in Algorithm 9. Also, we consider the resulted Π equals to the resulted optimal path or tree for respectively the unicast and multicast flows.

Algorithm 9 Pseudo-algorithm of the Batch Flow for CamCube Routing Protocol (`batch-(M)CRP`)

Inputs: $N(\mathcal{G})$, $L(\mathcal{G})$,B_i,$\mathcal{F}$ (Batch flow)
Output: Path/Tree Π for each admitted flow $\mathcal{F}_i$ (i.e.,$\mathcal{A}^{\mathcal{F}_i} = 1$)
ILP $\leftarrow$ MILP as in Problem $5 every \Delta_t$
OBJECTIVEOF(ILP) $\leftarrow$ **max** $f_1(x) = \sum_{\mathcal{F}_i \in \mathcal{F}} \mathcal{A}^{\mathcal{F}_i}$
Solve ILP to get optimal objective value $f_1^{(opt)}$
Push $f_1 \leq f_1^{(opt)}$ onto CONSTRAINTSOF(ILP)
OBJECTIVEOF(ILP) $\leftarrow$ **max** $f_2'(x) = z$
Solve ILP to get optimal objective value $f_2^{(opt)}$
Push $f_2' \leq f_2'^{(opt)}$ onto CONSTRAINTSOF(ILP)
OBJECTIVEOF(ILP) $\leftarrow$ **max** $f_3(x) = \sum_{l_{n1}^{n2}} (-x_{n1}^{n2}) \mid \forall l_{n1}^{n2} \in L(\mathcal{G})$
Solve ILP
for each $\mathcal{F}_i \in \mathcal{F}$ **do**
  if $\mathcal{A}^{\mathcal{F}_i} == 1$ **then**
    construct $\Pi^{\mathcal{F}_i}$ according to equation 7.4.
  end if
end for
Installation of the path/tree Π via SDN Controller(ONOS)

For the purpose of evaluation, if the algorithm cannot find a routing path/tree for any flow, we considerate it as rejected and we eliminate it from the waiting queue.

In the next section, we will describe the evaluation scenario by giving more details about inputs values.

7.4 Evaluation of Experimental Results

In this section, we tackle the performance of batch-(M)CRP proposal within CamCube DCN using SDN controller. We perform several experiments by emulating the environment detailed in Chapter 4. Next, we discuss the obtained results obtained by our proposal in comparison with the traditional shortest path routing based on batch processing within CamCube and Clos topologies. Also, as a first step, we tackle the performance evaluation for unicast batched flows by addressing UDP communication mode. Then, we finalize the testbed analysis by addressing the multicast batch processing flows through UDP communication mode.

Note that in order to simplify the performance evaluation we denote batch-CRP and batch-MCRP to define our proposed routing algorithm based on batch processing respectively for treating unicast and multicast flows. Besides, since our problem relies on 3 stages to resolve the main objectives of the proposal, we fixed the time limit for the first stage 1800 s, the time limit for the second stage to 600 s and the last stage to 300 s in order to get the resolution results in a reasonable time.

7.4.1 Performance evaluation for UDP Unicast traffic communication

In this section, we analyze the obtained results related to Latency, Packet Loss and Jitter to asses the efficiency of our proposal batch-(M)CRP in comparison with SPR and OSPF within CamCube and Clos DCNs, respectively. Then, we make a full comparison between the aforementioned protocols by varying traffic density.

7.4.1.1 QoS analysis

In Figure 7.1, we illustrate the latency ($\mathbb{L}$) of flows for both protocols. As shown in the above figure, it is straight forward to see that batch-CRP deeply minimizes $\mathbb{L}$ values compared to SPR and OSPF. So, batch-CRP generates latency values varying between 1.87 ± 1.47 ms for 100 servers and decreases to reach 0.32 ± 0.11 ms for 240 servers. Remember that our proposal performs an admission flows control for each batched flows by maximizing the acceptance while maximizing the residual bandwidth in the network. Thus, despite the limited number of rejected flows, we can deduce that batch-CRP provides a better QoS in terms of latency.
But, results show that SPR provides a high latency values for all topology size in comparison with OSPF and batch-CRP. We can explain the latter behavior by the fact that SPR cannot manage correctly a high number of arrived flows simultaneously. Since the latter protocol relies only on shortest path for routing, SPR strategy exploits the same links to route packets for different flows soliciting the same nodes in their paths simultaneously. Besides, in CamCube DCN, SPR protocol makes use of a higher number of switches compared to Clos topology which explains the obtained latency for the different strategies. For instance, the latency is equal to 627 ± 106 ms and decreases to reach 570 ± 64 ms respectively fo 100 and 220 servers, whereas, $\mathbb{L}$ is equal to 172.86 ± 5.79 ms and decreases to reach 139.19 ± 14.40 ms respectively for 100 and 220 servers. By the end, we deduce that the overall strategies provide a lower latency values when the topology size is increasing. Hence, the above behavior shows that the overall strategies exploits more

Figure 7.1 – UDP-Batch-CRP- Latency - $\mathbb{L}$

available links to route packets for an increased topology sizes which impact the latency values.

Consequently, these results show that our proposal manages better the arrived traffic than SPR and OSPF by capitalizing on the less congested links to transmit packet in spite of the increasing path length in small sized infrastructures i.e., until 240 servers.

Figure 7.2 depicts the IP packet loss ($\mathbb{P}$) for UDP unicast flows with respect to the dimension of CamCube and Clos DCNs. It is noteworthy to see that batch-CRP significantly minimizes the packet loss rate where $\mathbb{P}$ is equal to almost zero for all CamCube DCN sizes. However, as illustrated in Figure 7.2, SPR shows a small augmentation of $\mathbb{P}$ values less than OSPF protocol within Clos topology when we compare the latters protocols with batch-CRP behavior. Indeed, $\mathbb{P}$ for SPR is equal to $4.7 \pm 0.8\%$ for 100 servers and decreases to reach $3.5 \pm 0.4\%$ for 240 servers, whereas, OSPF strategy increases by up to 4 times the packet loss rate values compared with SPR i.e., $\mathbb{P}$ equals to $17.4 \pm 2.9\%$ for 100 servers and $11 \pm 2.6\%$ for 240. Then, we deduce that the aforementioned routing strategies provide almost the same results behavior for Online UDP flows and batched processing flows within CamCube and Clos topologies particularly for packet loss and jitter. This can be explained by the fact that the overall strategies manage similarly the treated traffic within the DCNs despite the different flows scheduling i.e., batched versus online mode.
Besides, these results assess the efficiency of batch-CRP in flows treatment by deploying paths within less congested links to forward packets to the destination in a high loaded network. Also, SPR and OSPF exploit more available links for an increased topology size which explains the decrease of $\mathbb{P}$ values for these topologies sizes. Note that, as explained in Chapter 5 and Chapter

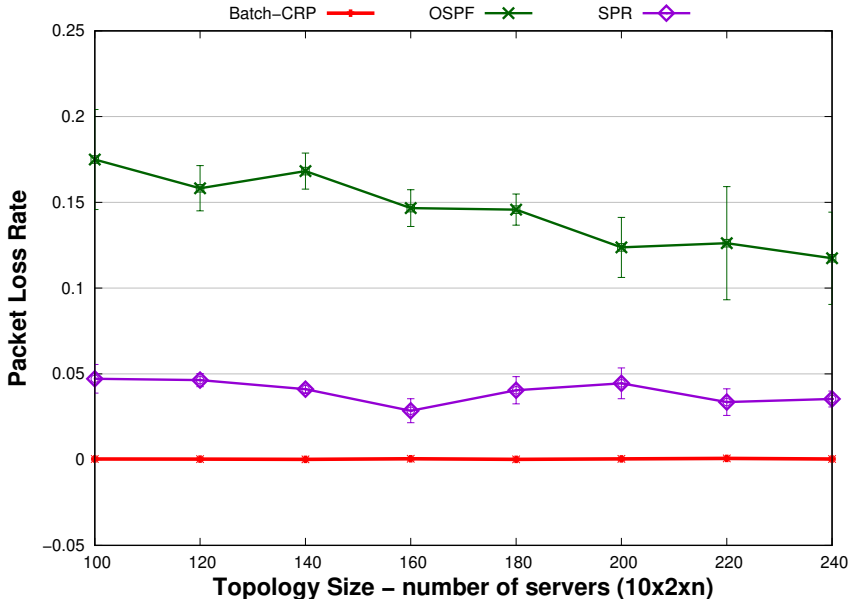

Figure 7.2 – UDP-Batch-CRP- Packet Loss Rate - $\mathbb{P}$

6, the variation of point in the figure, showing an unexpected behavior, is caused by the limited number of experimentations that we set to 5. Also, $\mathbb{RE}$ can confirm these values as illustrated in Table. 7.1, where RE(CR,SP) is equals to $-276,192$ and $-997,493$ within 180 servers respectively for SPR and OSPF.

Table 7.1 – Summary of the RE_Loss comparison for Batch processing flows in CamCube/Clos DCN

Number of servers	RE(CR,SP)	RE(CR,OS)
160	-50,478	-263,724
180	-276,192	-997,493
200	-98,785	-276,433

Figure 7.3 shows the obtained jitter ($\mathbb{J}$) for the studied protocols. As we can see, the CamCube-based protocols i.e., SPR and batch-CRP deeply minimize the jitter values by registering almost zero for all CamCube topology size. However, OSPF protocol, performing in Clos topology, increases $\mathbb{J}$ by 10 compared to SPR in CamCube DCN. For instance, the jitter registers a value of 9.12 ± 1.20 ms for 120 servers and start slightly decreasing until getting closer to the CamCube-based routing protocols. Note that the unexpected values shown in OSPF curve can be also explained as packet loss rate for the limited number of experimentation where Table. 7.2 shows RE(CR,OS) equals to $-1035,438$ for 180 servers. The aforementioned behavior of OSPF protocol

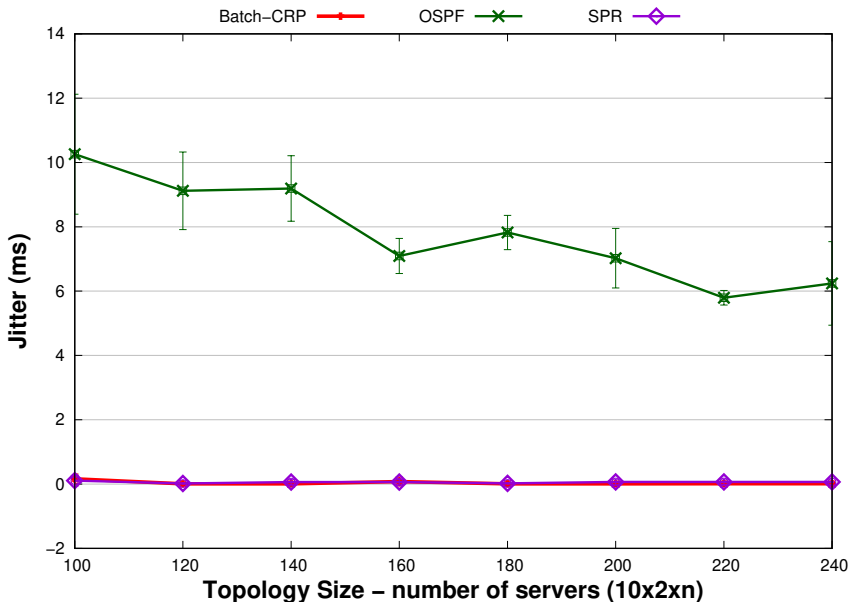

Figure 7.3 – UDP-Batch-CRP- Jitter - 𝕁

can be explained by the fact that Clos switches cannot manage well the increased traffic load i.e., when an important number of flows arrives at the same time for the treatment. Also, servers performing switch roles within CamCube topology provide better performance than ToR and Spine switches in Clos topology due to their important queue size and their high performance characteristics. However, for an increased Clos DCN size, the number of switches is increased. Consequently, the number of available links is also increasing. By doing so, the aforementioned behavior impact jitter values which are getting reduced from 100 to 240 servers for OSPF protocol. Besides, we can deduce that both SPR and batch-CRP provide a very tiny and stable jitter values despite of the increased CamCube size thanks to the high number of switches and multi-path routing within the topology.

Table 7.2 – Summary of RE_Jitter comparison for Batch processing flows in CamCube/Clos DCN

Number of servers	RE(CR,SP)	RE(CR,OS)
160	0,198	-88,425
180	-1,966	-1035,438
200	-6,856	-825,952

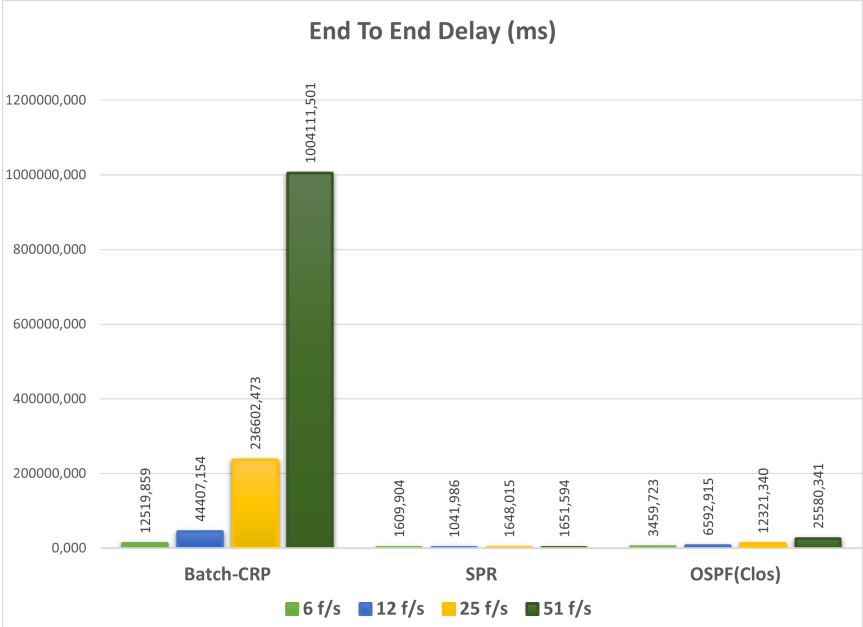

Figure 7.4 – UDP-Batch-CRP: Average of E2E Delay for different λ_f

7.4.1.2 Analysis for varied traffic density

In this section, we analyze the performance of batch-CRP protocol based on unicast flows with SPR and OSPF within 100 servers in terms of E2E delay and QoS for varied traffic density.

Figure 7.4 illustrates the E2E delay time including the convergence time of each strategy within 100 servers. It straight forward to see that batch-CRP shows a very high E2E delay that increases respecting the augmentation of the traffic load density compared to the shortest-path-based routing protocols. For instance, the E2E delay time is equal to 12.5 s for $\lambda_f = 6$ f/s and reaches almost 16 min to solve the traffic density $\lambda_f = 51$ f/s. Hence, batch-CRP generates a very long convergence time for batch processing flows which can deeply impact the QoS of cloud services in a real environment. The aforementioned behavior can be explained by the fact that batch-CRP relies on Cplex solver to provide the optimal solution for the set of arrived flows. Also, the latter solver takes a long convergence time in order to find the optimal solution through a high number of paths especially when it aims to maximize both residual bandwidth and flows acceptance in the system meanwhile. **To overcome it, as short-term perspective, we suggest to implement an heuristic algorithm that can provide a near to optimal solution while minimizing the overall processing time of the routing protocol** .
However, as we can see SPR strategy deeply minimizes the E2E delay time compared with OSPF scheme. Indeed, the latter protocol shows an E2E more than 6 times compared to SPR, i.e., 1.04 s and 6.5 s for $\lambda_f = 25$ f/s respectively for SPR and OSPF protocols.
Despite the minimized values of E2E delay for SPR protocol within CamCube DCN, we can deduce that all cited protocols can not manage the increased traffic density in a reasonable time

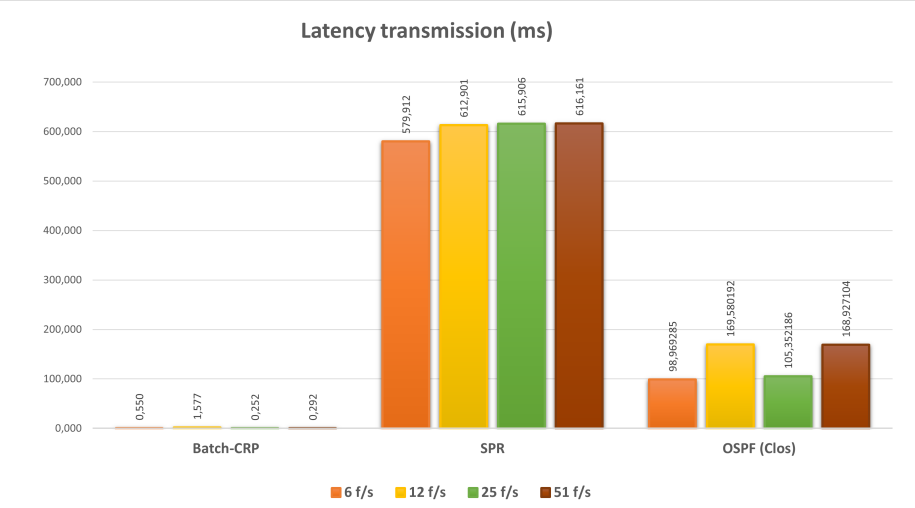

Figure 7.5 – UDP-Qos-Batch-CRP: Average Latency for different λ_f

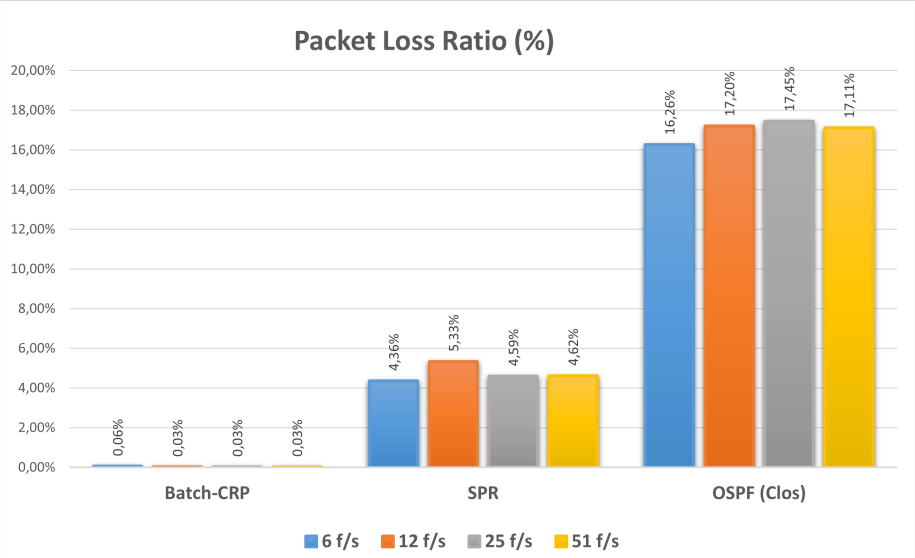

Figure 7.6 – UDP-QoS-Batch-CRP: Average Packet loss Ratio for different λ_f

within a real environment deployment. In fact, an average of $1-6$ *sec* to treat 25 flows per sec still important for a client within a real cloud environment characterized by thousands of flows. **So, the idea of implementing an heuristic may provide a solution to that issue by providing better performance than the existing strategy for one hand. Then, for a second hand, the heuristic can generate a complimentary performance to our** `batch-CRP` **by enhancing its convergence time while maintaining its objectives related to the maximization of both residual bandwidth in the network and the acceptance of batched flows.**

In spite of the very high convergence time generated by batch-CRP, the latter routing protocol provides a good QoS performance in terms of latency and packets loss rate as shown respectively in Figures 7.5 and 7.6 when varying the traffic density within 100 sized topologies. Indeed, as illustrated in Figure 7.5, batch-CRP provides a minimized latency values equal to 0.5 ms and that slightly decreases to reach 0.2 ms for $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s, respectively.
Furthermore, it is noteworthy to see that SPR highly increases the latency values compared to OSPF protocol within Clos topology and our proposal. In fact, for SPR, the latency values are equal to 579 ms and 616 ms for $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s, whereas, OSPF shows latency values equal to 98.96 ms and increases to reach 168.9 ms respectively for $\lambda_f = 6$ f/s and $\lambda_f = 51$ f/s. The above behavior of SPR can be explained by the fact that the latter protocol can reach a high bottleneck when managing a number of flows simultaneously. In fact, despite the high number of paths provided for CamCube servers, SPR can provide a same path between two nodes simultaneously solicited by different arrived flows in the networks. Thus, the congestion is maximized which can impact the latency values. Besides, for clos topology, since packets are transmitted at most via two switches (ToR and Spine) for an inter-rack transmission, the switches can quickly become congested and more packets will be dropped. But the latency for OSPF still relatively lower than SPR because of the limited number of traveled hops (switches) and the fewer number of treated flows in Clos topology. Last but not least, results show that the latency values are increased according to the augmented traffic load which can be explained by the augmentation of congested links and switches accordingly.
Furthermore, considering the packet loss rate, batch-CRP deeply minimizes the packet loss rate $\mathbb{P}$ and shows a stable value equals to 0.03% despite the increasing values of λ_f. We consider that the stable obtained results show a non congested system and the emulated scenario cannot occupy the total links bandwidth which makes our proposal keeping the same behavior despite the different traffic densities values. However, OSPF shows more than 4 times $\mathbb{P}$ values for SPR i.e., $\mathbb{P}$ equals to 4.36% and 16.26% respectively for SPR and OSPF protocols when $\lambda_f = 6$ f/s. This can explained that the increasing number of treated flows at the same time i.e., batch processing flows, can impact the performance of Clos switches. Indeed, according to the obtained results for a raised traffic density, Clos switches suffer from congestion and hence reject flows packets accordingly.
These obtained results show that routing strategies within CamCube DCN can provide better performance in terms of packet loss in respect to the increased traffic density compared to OSPF protocol within Clos topology. This can be explained by the fact that SPR and batch-CRP can manage better the exploitation of the available paths and takes profit of the multipath routing presented by the CamCube topology in order to achieve a good QoS in terms of packet loss.

To conclude, our proposals based on unicast flows CRP and batch-CRP keep almost same behavior by minimizing the latency and the packet loss rate independently of the flow scheduling mode and despite the presence of the admission control within batch-CRP. However, for SPR and OSPF based on shortest path strategy, both routing schemes suffer from a high latency values and rejects more packets. Particularly, for a batch processing flows, we notice that despite the acceptance of 100% of arrived flows for treatment (opposite to batch-CRP mechanism), the different flows cannot be transmitted correctly especially if they exploited common link to transmit packets in the network.

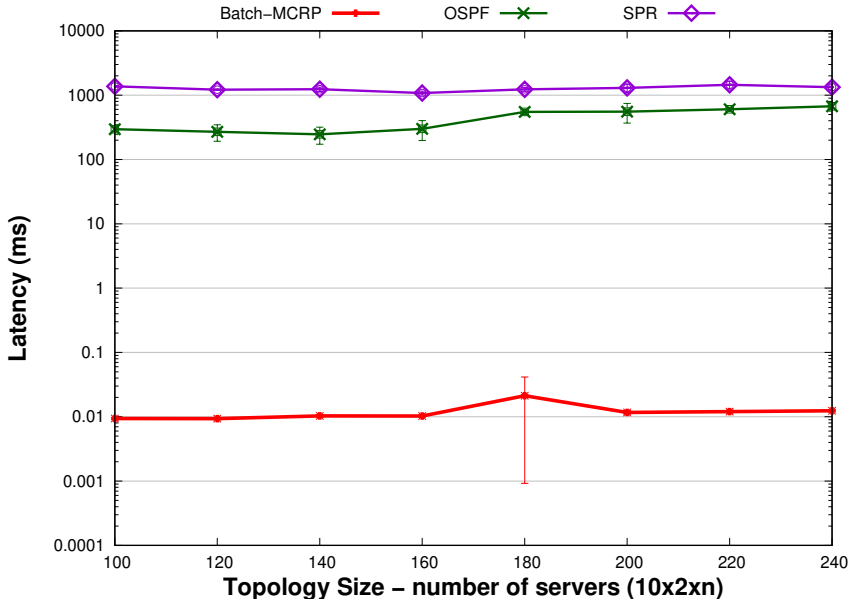

Figure 7.7 – UDP-Batch-MCRP-Latency - $\mathbb{L}$

7.4.2 Performance evaluation for UDP Multicast traffic communication

In this section, we will evaluate the performance of batch-MCRP in comparison with SPR and OSPF schemes. The evaluation is based on comparing the performance of the above strategies in terms of E2E Delay and QoS (Latency and packet loss) by varying both topology sizes and traffic density within the DCNs.

7.4.2.1 QoS analysis

In this section, we evaluate the several protocols by varying the CamCube and Clos dimensions while keeping a constant $\lambda_f = 12$ f/s. Then, we analyze the obtained results in terms of quality of multicast tree and QoS i.e., latency, packet loss rate and jitter.

In Figure 7.7, we illustrate the obtained latency for the different routing scheme.

It is straight forward to see that our proposal (batch-MCRP) generates an optimized multicast trees with a minimized latency $\mathbb{L}$ values starting from 0.009 ± 0.0003 ms in 100 servers to 0.012 ± 0.0004 ms in 240 CamCube servers. Thus, for an increased topology size (by 24 times), batch-MCRP shows multicast flows latency increased by 10. This can be explained by the fact that our proposal exploits more links to transport traffic. Hence, the number of intermediate nodes is increased to build the multicast tree which impacts the latency.

Considering SPR and OSPF protocols, we notice that both strategies perform almost the same behavior for batched flows independently to the communication mode (unicast and/or multicast) by maximizing the latency. Particularly, SPR strategy illustrates that latency equals to 1372 ± 29 ms in 100 and decreases to reach 1339 ± 84 ms in 240 CamCube server. This high

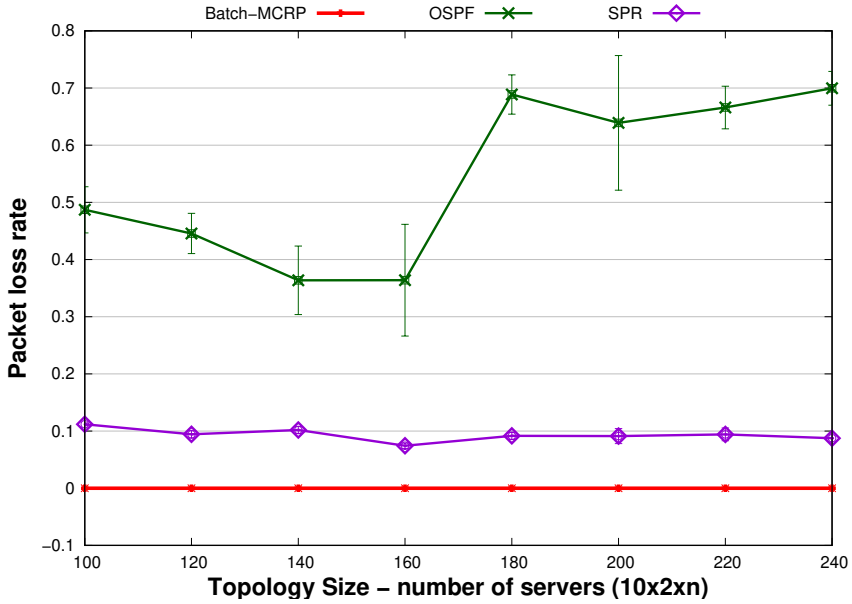

Figure 7.8 – UDP-Batch-MCRP-Packet Loss Rate - $\mathbb{P}$

value of latency is explained by the presence of highly congested links in the CamCube DCN despite the multiple switches compared to Clos topology. Indeed, SPR makes a long time to generate the routing path for the concurrent flows arriving simultaneously. Also, in opposite to the unicast mode, SPR should provide different paths to reach the multicast destinations for a given source which increases the concurrence between flows especially when they solicit the same nodes to transmit traffic.
Furthermore, for OSPF protocol, $\mathbb{L}$ values are equal to 304.35 ± 43 ms and 708.50 ± 110 ms respectively with 100 and 240 servers. The above considerable latency values can be explained by the presence of limited number of switches to treat the high number of arrived flows at the same time.
By the end, results show that SPR and OSPF admit 100% of arrived batched flows. However, the latter protocols suffer from high congestion since they solicit mostly the same links/switches to route packets which enhance the transmission delay expressed by the latency values as shown in Figure 7.7. But, by deploying an admission control, we notice that batch-CRP and batch-MCRP succeed to make a trade-off between maximizing the number of treated flows and ensuring the load balancing in the CamCube network. Hence, our proposals provide a better performance in term of latency in a batched flows context.

Figure 7.8 depicts the packet loss rate ($\mathbb{P}$) for batch processing multicast flows. It is straight forward to see that batch-MCRP highly decreases values and it is equal to zero independently to the increased CamCube size. Moreover, SPR provides a higher $\mathbb{P}$ values compared to batch-MCRP and still minimized compared to OSPF scheme within Clos architecture. Also, it is noteworthy to see that despite the approximately constant behaviour of SPR packet loss values, $\mathbb{P}$

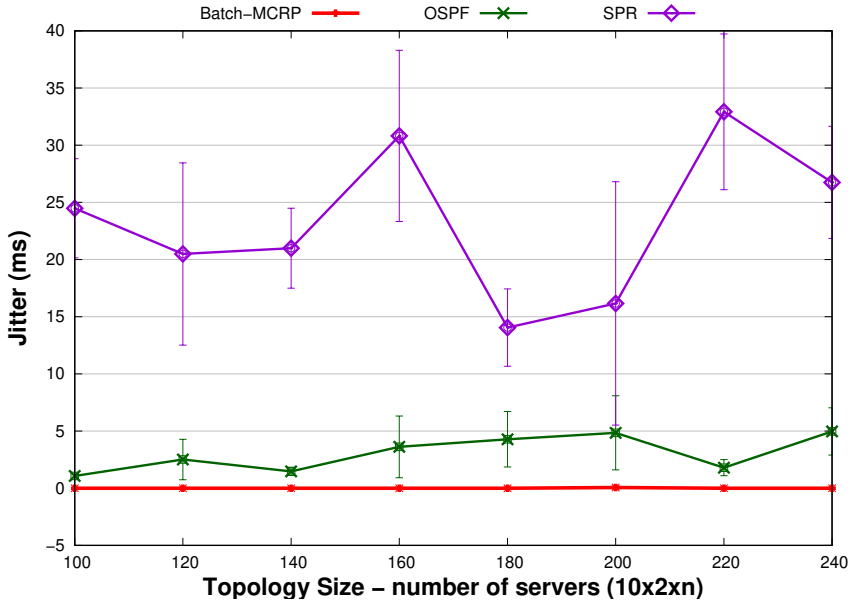

Figure 7.9 – UDP-Batch-MCRP-Jitter - $\mathbb{J}$

is progressively increasing with respect to the increased Clos DCN dimensions. Indeed, OSPF registers $\mathbb{P}$ values up to 4 times greater than SPR i.e., $11 \pm 0.002\%$ versus $49 \pm 0.04\%$ within 100 sized topology. The augmentation of packet loss rate when performing OSPF strategy within Clos topology can be explained by the limited number of ToR and Spine switches to handle with the congested network.

Figure 7.9 illustrates the jitter ($\mathbb{J}$) values for batch-MCRP compared to SPR in CamCube DCN and OSPF in Clos topology. It is straight forward to notice that the jitter is deeply minimized within batch-MCRP. Also, it is undeniable that, in addition to the latency and packet loss rate, our proposal outperforms the different strategies in term of jitter i.e., $\mathbb{J}$ is equals to 0.001 ms for all CamCube topology sizes. But, SPR strategy presents a high values of jitter compared to OSPF and batch-MCRP. In fact, $\mathbb{J}$ reaches a maximum value equals to 33.71 ± 5.50 ms within 220 servers and a minimum value equals to 14.20 ± 2.64 ms in 180 servers for SPR protocol. Hence, a difference of 19.5 ms. This considerable variation can be explained by the limited number of experimentations and the fast congested switches to handle the high traffic load particularly for installing the multicast tree for the batch flows following the shortest path. Indeed, despite the increased number of switches in CamCube compared to the Clos switches, SPR makes use of congested links since it relies on the shortest path routing which can increase the traffic bottleneck. Hence, the jitter is increased.

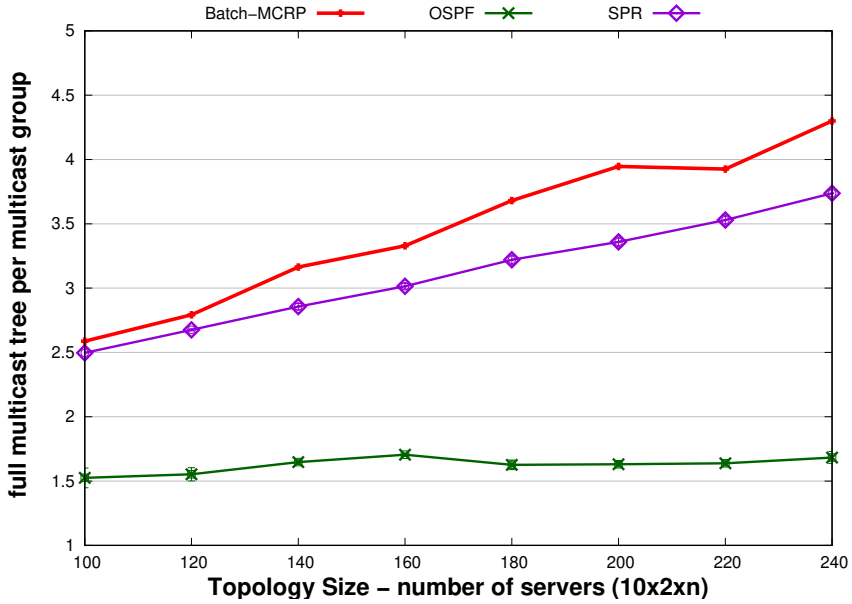

Figure 7.10 – UDP-Batch-MCRP-Average Size ratio- Multicast Quality tree - $\mathbb{T}$

Figure 7.10 shows the quality tree presented by the ratio $\mathbb{T}$ of Batched multicast flows within CamCube and Clos DCNs.
It is noteworthy to see that OSPF minimizes the number of nodes within multicast tree for Clos topology compared with protocols deployed in CamCube. In fact, Figure 7.10 shows that averaged $\mathbb{T}$ for OSPF is varying from 1.5 to 1.7 respectively for 100 and 160 servers. This can be explained by the fact that for intra-rack communication, servers reach their destinations via ToR switch and for inter-rack communication, servers can solicit at most 2 Clos switches to reach their destinations i.e., ToR and spine switches. Therefore, the curve of $\mathbb{T}$ ratio still constant despite the increasing number of topologies servers.
However, for CamCube topology, we notice that SPR and batch-MCRP can progressively increase with respect to the augmentation of the DCN size. However, batch-MCRP shows a highest $\mathbb{T}$ values i.e., where the number of nodes register up to 4 times the multicast group for 240. A maximum $\mathbb{T}$ values equal to 4.27. The above behaviour explained by the fact that batch-MCRP relies on Cplex to find the optimal solution in order to maximize the residual bandwidth in the network, then an increased number of available link. Also, CamCube topology is known by its long routing diameter which explains the increased $\mathbb{T}$ for SPF despite the use of the shortest path to reach the destinations.
To conclude, our proposal presents a dense multicast tree to provide a better bandwidth utilization in the network. Also, batch-MCRP exploits more links to overcome the congestion issues when it handles the high traffic engendered by the treatment of high number of flows at the same time.

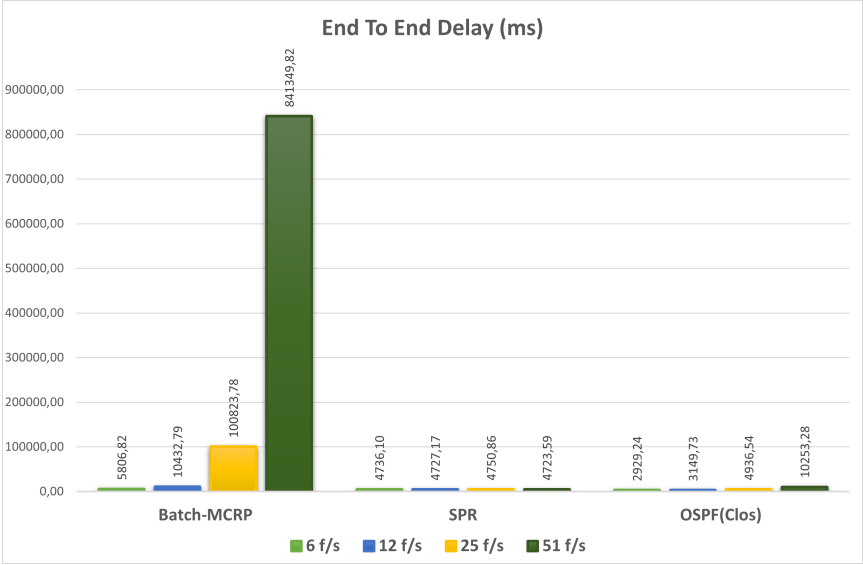

Figure 7.11 – UDP-Batch-MCRP:Average of E2E Delay for different λ_f

7.4.2.2 Analysis for varied traffic density

Figure 7.11 illustrates the E2E delay for the studied strategies. It is straight forward to see that Bath-MCRP provides a very high E2E delay that increases following the traffic load augmentation. Indeed, the E2E delay of batch-MCRP starts from an average of 12.5 s for $\lambda_f = 6$ f/s to an average of 16.57 min for a maximized density equals to $\lambda_f = 51$ f/s. This high convergence and transmission time can be explained by the use of Cplex solver for batch-MCRP to generate the optimal solution for the high number of flows treatment at the same time. Also, batch-MCRP aims to maximize the acceptance of batched arrived flows which can impact the overall resolution time for the generation of multicast trees. Therefore, similarly to batch-CRP, **we suggest to propose a faster resolution e.g. based on meta-heuristic algorithms in order to provide a multicast tree for each arrived flow in a reasonable time** .
Regarding SPR protocol within CamCube topology as batch-MCRP, we can see that the shortest path strategy provides a progressive E2E delay augmentation according to the augmentation of the CamCube network load. However, this augmentation of E2E delay time for SPR is tiny despite the increase of traffic density. For instance, when λ_f is doubled from 25 f/s to 51 f/s, the E2E delay is equal to 1648 ms and 1651.6 ms, respectively. We believe that for SPR resolution the deployed scenario cannot generate a high number of concurrent flows, so that SPR can handle better the congestion of the CamCube network. Hence, the E2E delay is minimized compared to batch-MCRP.
Furthermore, compared to SPR, OSPF strategy within Clos topology provides a doubled E2E delay i.e., 3459 ms versus 1609 ms when $\lambda_f = 6$ f/s, and when $\lambda_f = 51$ f/s, the E2E delay for OSPF can reach up to 15 times the E2E delay for SPR. The aforementioned behavior can be explained by the fact that limited number of Clos switches can rapidly get converged when the traffic load is increasing compared to the SPF within the voluminous and high performed switches.

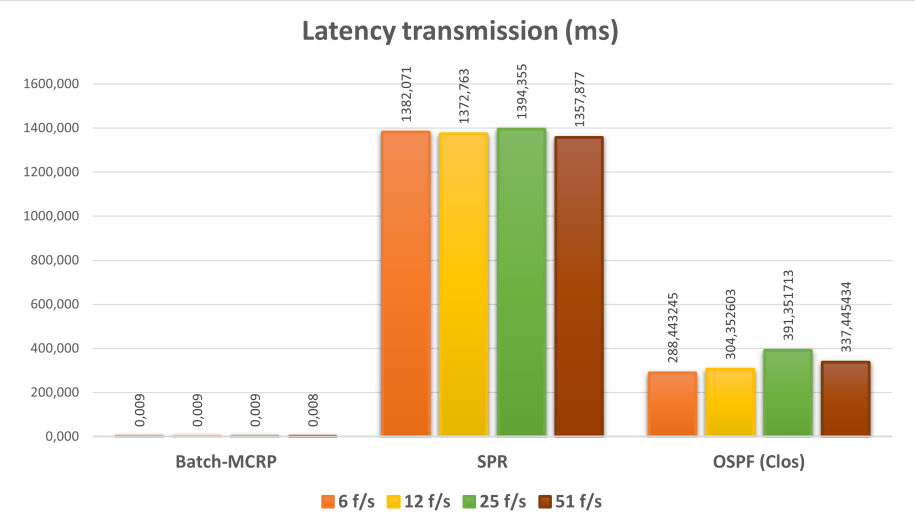

Figure 7.12 – UDP-Qos-Batch-MCRP: Average Latency for different λ_f

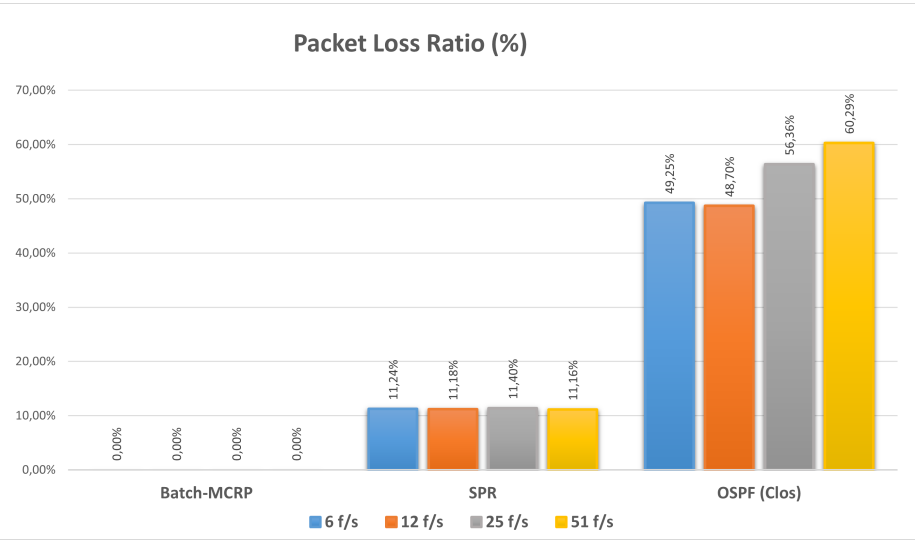

Figure 7.13 – UDP-QoS-Batch-MCRP: Average Packet loss Ratio for different λ_f

To conclude, protocols based on shortest path e.g., OSPF and /or SPR can provide a relatively better E2E delay compared to our proposal. In the next figures analysis, we suggest to evaluate the performance of these strategies in terms of QoS when we boost the traffic density in the system.

Figure 7.12 depicts the latency values generated by the different strategies in CamCube and Clos DCNs. Results show that batch-MCRP achieves the lowest latency values compared to SPR and OSPF i.e., almost same values for a doubled traffic density from $\lambda_f = 25$ f/s to $\lambda_f = 51$ f/s, $\mathbb{L}$ equals to 0.009 ms. This behavior can be explained by the fact that batch-MCRP makes use of

Cplex solver to find the optimal solution. So, despite the long convergence time, batch-MCRP provides a better utilization of available links which impact the rapidity of transmission within the obtained multicast tree. Also, Figure 7.13 shows that batch-MCRP minimizes the packet loss rate for the same reason and record zero despite the increased traffic load. Hence, we can conclude that despite its long convergence time, batch-MCRP can deeply minimize the latency and the packet loss rate. Then, our proposal provides a better quality of services for batched arrived multicast flows. However, regarding the obtained latency values for SPR and OSPF, we can deduce that SPR shows a very high latency values compared to OSPF. This can be explained by the fact that for the different traffic load, SPR protocol suffers from congestion which can impact the transmission delay for concurrent flows soliciting common links. Also, the high number of travelled nodes in CamCube topology can impact the transmission packets delay for SPR protocol. Hence, the value of latency is increasing in respect to the high traffic load. Besides, for Clos topology, the latency still higher than batch-MCRP because of the limited number of switches responsible for the transmission of multicast flows within Clos topology. For instance, concurrent flows, treated by OSPF strategy, can solicit maximum 3 switches (e.g., ToR1 - Spine1- ToR2) to route their packets for intra-rack or inter-rack communication. Consequently, the switch nodes can quickly suffer from bottleneck especially in case of traffic load augmentation which can also impacts the packet loss.
Indeed, packet loss results in Figure 7.13 can confirm the aforementioned behavior. For instance, OSPF protocol packet loss rate augments when the traffic load is increased i.e., $\mathbb{P}$ equals to 49% for $\lambda_f = 6$ f/s and reach 60.2% for $\lambda_f = 51$ f/s, whereas, $\mathbb{P}$ for SPR records only 11% for the different λ_f values. In fact, we can explain that for SPR protocol takes a long time to transmit packets which impacts the latency values. But, thanks to the multiple number of links/switches characterizing CamCube topology, the dropped packets are limited when transmitting multicast flows for SPR routing scheme.
Consequently, we can deduce that shortest-path protocol shows a short convergence time but engender a degradation of QoS in terms of latency packet loss rate. However, our proposal shows a better QoS performance independently of the traffic load density and communication mode i.e., unicast or multicast batched flows.

Note that we are running limited number of experimentations which can provide less confident values. Thus, this can explain constant behavior of the studied strategies in some figures.

7.5 Conclusion

In this chapter, we optimized the routing of flows in batch mode for CamCube topology. To do so, first we emulated CamCube server only DCN infrastructure based on ONOS SDN controller and Mininet emulator. Then, we proposed a new algorithm `batch-(M)CRP` to tackle the batch routing problem for both unicast and multicast flows. We compared the obtained results of our proposal with the traditional shortest path within CamCube and Clos DCNs.
The experimental results show that our proposal achieves good quality of service performance. Despite of achieving the design goals, we assume that solving the batch arrived flows via CPLEX has a strong influence on the convergence time. Also, it is noteworthy to mention that by our proposal, we aim to provide a **preliminary study** for batch arrived flows within CamCube DCN where we seek to evaluate its feasibility in a such environment. Hence, we consider that the obtained results are prominent but not realistic, particularly for a resolution time set to $45\ min$.
Consequently, as a future work, we suggest to improve the performance of our proposal in term of convergence time by solving it through heuristic solutions for a high number of batch arrived flows. Also, in order to improve the performance analysis of our proposal, we propose to compare `batch-(M)CRP` protocol to other strategies different from shortest-path based protocols.

Chapter 8
Conclusion

Contents

8.1 Introduction

In this chapter, we will conclude this thesis. In Section 8.2, we will summarize our proposals. Afterwards, in Section 8.3, we will provide some short and mid term perspectives. Finally, in section 8.4, we will enumerate a list of our accomplished publications in this thesis.

8.2 Summary of Contributions

During this thesis, we focused our research on the routing and resource allocation optimization problem for CamCube server-only topology. Particularly, our main target consisted on dealing with scalability issues while maximizing the QoS of intra CamCube DCN flows. To do so, we propound several routing protocols through extensive emulations in order to study the efficiency of our proposals in a real-time environments. Hereafter, we provide the main contributions:

- In Chapter 4, we detailed our proposed architecture and platform used in this thesis to evaluate the propounded algorithms. We presented the objective of our work and we described the deployed platform. To run the virtualized CamCube architecture, we used Mininet emulator. Then, we adopted ONOS SDN controller to manage the CamCube DCN traffic and to run our proposed algorithm in order to optimize routing and QoS within the CamCube network. In this chapter, we validated the proposed platform by deploying the main related routing protocol in CamCube DCN i.e., Shortest Path Routing Algorithm.

- In the first contribution, detailed in Chapter 5, we tackled the unicast communication flows for CamCube servers managed by ONOS SDN controller. In this context, we formulated CamCube routing problem as a Mixed Integer Linear Programming model and we proposed `CRP` scheme based on Branch and Cut algorithm to solve it. Our proposal sought to enhance the QoS of CamCube DCN flows by maximizing the residual bandwidth and minimizing the number of traveled hops in the optimized path. Besides, in order to decrease the convergence time comparing to results obtained by `CRP`, we implemented a new unicast protocol `ACO-CRP` based on Ant Colony meta-heuristic. Afterwards, we compared the obtained results with the shortest path within CamCube and Clos DCN in order to evaluate the QoS performance of our proposals `CRP` and `ACO-CRP` in terms of latency, packet loss rate and jitter. Moreover, we compare and discuss the performance of the aforementioned protocols by varying the traffic density. It is noteworthy to say that we analyzed the obtained results for both UDP and TCP flows communication.

- In Chapter 6, we developed the second contribution where we addressed the multicast routing in CamCube server-only DCN by proposing `M-CRP` protocol. The latter aimed to maximize the residual bandwidth in the network while minimizing the number of hops. In this regard, we emulated the CamCube DCN topology with Mininet and we synchronized it with ONOS controller. Also, `M-CRP` protocol is formulated as a lexicographic multi-objective optimization problem. Later, in order to enhance the convergence time of `M-CRP`, we propose a meta-heuristic model based on Ant Colony Optimization for multicast flows, named `ACO-MCRP`. Afterwards, we compared our proposals to the shortest path and

OSPF protocols respectively for CamCube and Clos topologies in terms of network QoS, quality of the obtained multicast tree. Finally, we deduced that M-CRP and ACO-MCRP proposals show better results than the shortest path in terms of QoS (i.e., latency, jitter and packet loss) and multicast tree quality (i.e., depth of the tree).

- In the third contribution detailed in Chapter 7, we focused our study on batch communication mode within CamCube data center. Despite the important results obtained in the aforementioned contributions, we considered that deploying such a topology using SDN controller can generate a QoS deterioration (e.g., congestion engendering the loss and/or rejection of arrived flows). In this context, we discussed the flows arrival through batch mode. Then, we propounded batch-(M)CRP protocol based on lexicographic multi-objective problem that aimed to maximize the QoS of the network. We emulated the implemented platform with Mininet and ONOS controller to study the efficiency of our proposal. Finally, we discussed the obtained results for UDP unicast and UDP multicast flows communications following the batch mode. Besides, we varied the traffic density for both communications schemes and we made a full comparison between the shortest path, batch-(M)CRP in CamCube DCN and OSPF within Clos DCN. Finally, we conclude that batch-(M)CRP protocol provides better performance than the shortest path protocol within CamCube DCN and OSPF for Clos DCN.

8.3 Perspectives

In this section, we point out some open research perspectives of our work that can be the subject of future research projects. In fact, we suggest to classify them into *short-term* and *mid-term* perspectives as follow:

For **the short-term** perspectives, first, it is straightforward to see that our batch-(M)CRP proposal needs a long convergence time. So, we suggest to solve the proposed problem by heuristic algorithms such as [202] based on Monte Carlo tree search in order to study the batched arrived flows for both unicast and multicast communications. Second, we can enlarge our experimentation by comparing our proposals to existent routing schemes different than shortest-path-based protocols [140] [203]. Third, we suggest to enhance our propositions to take into account the difference between the transport layer protocols (e.g., TCP and UDP). Indeed, our proposals in this thesis does not consider the different mechanisms of TCP (e.g., congestion control) driven by the impact of both the network condition (e.g., propagation delay) and also the capacity constraints on both the client downloads and server uploads on throughput. Besides, we aim at investigating the advantages of other routing algorithms with different objective functions, e.g., minimizing the packet loss or the E2E delay. We can for instance compare different objective functions in terms of admission rate and throughput. Some works concluded that min-max fairness (a special scheme of alpha-fairness [204]) can model with high precising the sharing between the TCP flows within a network [205]. Thus, it is interesting to investigate this issue within CamCube topology where i) several TCP flows can often share several links, and ii) some servers may upload data in parallel to many other servers e.g., in the case of some flows do not request a predefined bandwidth in advance.

As **mid-term** perspectives, we suggest to move forward from intra-data center traffic managed by a single SDN controller to an inter-data center environment orchestrated by a cluster of SDN controller. In fact, as a future work, we propose to manage CamCube data centers geographically distributed by SDN cluster where we make use of our different optimization proposed algorithms in order to optimize the routing and resource allocation in this environment. Indeed, the objective of this work is to study the resilience and the efficiency of our proposals in distributed data center architecture and then to compare it to previous work presented in the literature such as [206] [207].
It would be interesting to add the instantaneous packet loss and also the residual bandwidth metrics (as explained in Chapter 4, section 4.4.1) as input to our optimization problem (e.g., constraints) in order to build an application QoS aware solution (with some predefined level of loss and throughput). In addition, Reinforcement Learning (RL) techniques have received a lot of attention for optimizing performance in networking domain. We believe that RL can help for building an adaptive resource allocation solution face to the network dynamics [208].
In addition to that, we can move from improving the cloud private IaaS to the hybrid cloud Paas and/or SaaS levels for study. In fact, we suggest to study the possibility to synchronize the proposed algorithms of this thesis with monitoring tools for multi-cloud platforms. Then, we propose to make this monitoring tools able to provide the needed customers solution e.g., through established multi-cloud SLA agreements [209] and QoS-functionality/price ratio optimization [210] [211] in order to deploy dynamic services levels in the cloud [212].

8.4 Publications

- Roua Touihri, Safwan Alwan, Abdulhalim Dandoush, Nadjib Aitsaadi and Cyril Veillon "SDN-Based Batch Flow Routing in CamCube Server-Only Data Center Networks", **published**, in IEEE International Conference on Communcations (ICC), Dublin, June 7-11, 2020.

- Roua Touihri, Safwan Alwan, Abdulhalim Dandoush, Nadjib Aitsaadi and Cyril Veillon "M-CRP: Novel Multicast SDN based Routing Scheme in CamCube Server-only Datacenter", **published**, in IEEE Global Communications Conference: Communications Software, Services and Multimedia Apps (GLOBECOM), Hawaii, December 9-14, 2019.

- Roua Touihri, Safwan Alwan, Abdulhalim Dandoush, Nadjib Aitsaadi and Cyril Veillon "CRP: Optimized SDN Routing Protocol in Server-Only CamCube Data-Center Networks", **published**, in IEEE International Conference on Communcations (ICC), Shanghai, May 20-24, 2019.

- Roua Touihri, Safwan Alwan, Abdulhalim Dandoush, Nadjib Aitsaadi and Cyril Veillon "Novel Optimized SDN Routing Scheme in CamCube Server Only Data Center Networks", **published**, in IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, January 11-14, 2019 (Poster).

Appendix

During my PhD, I was named as the referent IT of Devoteam Research and Innovation (DRI). Moreover, I was appointed as cloud/SDN/DevOps project manager where I leaded and was involved in several Cloud and IT projects within DRI presented from the oldest to the recent ones as follow:

1. ***Comparison between data centers simulators*** *2016*:

   The objective of this project was to make a full state of the art of the existing data center simulators in order to identify the most interesting simulator to test. The study was based on the complexity of deployment and the compatibility with the addressed DCNs. We believed that a data center simulator can help us to design efficiently a new deployed platform for DRI. By the end of this mission, the consultants realized a comparative study based on the aforementioned criteria. this study leads to two simulators: i)`GridSim` and ii) `CloudSim`. There was two consultants involved in this project where each one has the mission of studying, testing and evaluating one simulator. Finally, the comparison leaded to choose the commonly used `CloudSim`. However, this simulator makes use of Java language whereas our platform is based on python and consultant are more comfortable with the latter programming language. Also, the consultant encounters some difficulties to adapt this simulator to our DCN architecture (e.g., CamCube). Therefore, as a future work, we suggested to find an other simulator or emulator that can fulfill our technical objectives (simulation of the servers connection and links characteristics in respect with the DCN architecture) and that can easily be deployed in our environment. Later, we proposed to deploy Mininet as a solution.

2. ***DRI IT set up and deployment*** *2016-2017*:

   I had the opportunity to present a new private cloud design based on Openstack for DRI named `Devocloud`. The objective of this mission was to design and deploy a high-available data center architecture orchestrated by Openstack. The team was composed of two senior risk and security consultants, a cloud consultant and 2 cloud engineers for the setup and the installation of the newly designed platform. As a matter of fact, the implication of risk and security consultants was important in order to help us to develop a performed private cloud while respecting the security normalization and rules. Hence, to avoid any intrusion or security vulnerability that can badly impact the overall network of the company. My mission was to coordinate teams, to define the road map and to validate project steps and/or tasks through several technical and functional reports. Note that the coordination between teams and the respect of deadlines despite the unexpected events were the hardest tasks to deal with. But, it is noteworthy to say that the exchange with experienced consultants helped me to improve my management skills, particularly for the organisation of a project that is involving different members from several fields (risk and security, system and network administration, etc.).

3. ***OpenDayLight SDN controller deployment within OpenCif plateform*** *2016-2017*:

   After proposing the Devocloud platform, I suggested to implement and build a new OpenCif platform based on the same version of Openstack in order to make some tests and research studies without impacting the performance of DRI services provided by Devolcoud for our clients i.e., consultants working on research projects. The objective of this mission was to deploy an SDN controller in OpenCif platform to test a centralized network management within private cloud. In fact, We deactivated the Neutron[1] module and replaced it by the OpenDayLight SDN controller which was able to manage the Openstack network and to execute naturally the Neutron processes. Then, we concluded that OpenDayLight could replace the Openstack network module as the own native projects existing in the platform. It is noteworthy to say that the choice of OpenDayLight SDN controller was made after a full comparative study of the most relevant SDN controllers existing in the market at that time. Some of the comparative criteria were the compatibility with OpenStack and the facility of deployment. This project was carried out by a system and network engineer as part of its graduation project.

4. ***Private cloud Migration based on Openstack*** *2017-2018*:

   Victim of its success, the previous platform Devocloud was no more able to host the highly requested VMs and unable to support the new OpenStack version. Also, in order to ensure the platform stability and to have an IT support maintenance, we suggested migrating the previous platform to a new one by installing the latest Openstack RedHat version. Meantime, we aim to ensure a continuous and available service to our customers when migrating i.e., hot migration. Besides, our goal is to guarentee the availability of the hosted services, the ticketing support to answer the clients inquiries and the accessibility to the platform. To do so, we studied the feasibility of the migration and we made a road map respecting the time and financial constraints. The new platform Devolab was designed, implemented and deployed. In addition, the emerging architecture provided 61.8% of servers performance enhancement. Moreover, the project objectives were reached by providing a stable and maintainable platform. It is noteworthy to say that the hardest task was to ensure the availability of cloud services while migrating and the stability of the newly deployed servers to host the migrated VMs. This project was carried out by a cloud engineer as part of its graduation project. My role consists on supervising, validating the new design and evaluating task and steps of the project.

5. ***SONA deployment within OpenCif platform-based on Openstack and ONOS SDN controller*** *2018*:

   ONOS community had proposed the integration of their SDN controller within private cloud platform based on OpenStack through SONA project[2]. The newly designed SDN controller promises new features and more performed applications than its competitors i.e., OpenDayLight and Opencontrail. In this project, we proposed to test the installation

[1] Neutron is an Openstack SDN networking project responsible of delivering Network-as-a-service to the other Openstack projects/modules such as Nova Module.

[2] https://wiki.onosproject.org/display/ONOS/SONA+Architecture

of ONOS within our platform OpenCif to learn how ONOS can replace the Neutron project process within OpenStack. The results showed that ONOS was compatible with OpenStack architecture and offered a variety of applications via its northbound API that helped cloud administration to easily manage the OpenCif network, VMs networks and the physical infrastructure of the platform. The project was accomplished by two interns specialized in network and system administration. During their internship, the students installed ONOS within a VM hosted in a separated server and made the communication between ONOS and OpenCif platform based on OpenStack. For testing, they have deployed VMs within OpenStack and visualized the deployed virtual architecture in the graphic interface provided by ONOS. Thanks to this implementation, we were able to analyze the traffic generated by the VMs within OpenStack and their performance via the GUI provided by ONOS e.g., launching ping and/or iperf commands, visualizing the synchronization messages of the controller, etc. Since we deployed a customized architecture i.e., OpenCif, it was hard to elaborate the communication between ONOS and the latter platform especially when SONA project provides a general case of deployment and lacks of technical details for the implementation.

6. ***Development and Integration of CPMAN++ application in ONOS Northbound API*** *2018*:

   Furthermore, in order to evaluate and enhance the performance of our OpenCif platform in terms of QoS and then QoE, we suggested exploiting one of the most interesting applications provided by ONOS Northbound APIs called CPMAN. In fact, this application can provide real-time information and displays visual statistics about links, devices and topology states of the orchestrated data center according to administrator demand. Hence, the idea of this project was to propose an enhancement of this application and proposing CPMAN++ that was not only able to retrieve information about the physical devices of the data center topology but was also able to calculate the latency, the residual bandwidth and the packet loss rate via real-time information state. Then, we simulated a traditional data center topology managed by ONOS controller and we included the CPMAN++ to test it. The results showed that the developed application is able to retrieve the requested information and grabbed real-time statistics in terms of latency and packet loss rate. Actually, the developed application can be easily integrated into any installed ONOS controller. However, it still needs an improved display for a production use. The difficulty encountered in this project consists on the installation of ONOS from scratch. In fact, the intern who accomplished this work has encountered many bugs related to the installation of ONOS for several versions particularly when ensuring their compatibility of the existing OS and VM system performance. Note that the project was carried out as a part of graduation project for a network programming engineer.

7. ***Security Audit of DRI platforms*** *2019*:

   Once the different platforms were well deployed and became operational, we focused on the security area. In fact, we believe that despite their performance, the obtained platforms suffer from security weaknesses that we should take into a consideration to avoid any harmful intrusion. Therefore, we made a full risk and security audit and we planned several tasks for short and mid-term actions. The project was realized by two gradu-

ated engineers where they followed the referential "*Le guide d'hygiène Informatique de l'ANSSI*". The result of the study shows that the company has a green indicator (which is recommended) but still require some security enhancement. For instance, we suggest to start by securing the DRI services access by: i) defining different administrators profiles and right accesses, ii) adding certificates to the DRI services websites, iii) controlling access via passwords policy enforcement within the different equipments and iv) encrypting the hard disks of the different collaborators machines.

8. ***Integration of Kubernetes and ONOS in OpenCif Platform*** *2019*:

   The objective of this project was to integrate a cluster of Kubernetes in `OpenCif` platform based on OpenStack and to study QoS performance of the communication between a Kubernetes pod and an installed OpenStack VM. We started by using `OpenCif` and deploying a containerized platform based on Kubernetes installed in a different compute node. Then, we addressed the communication between the Openstack VM and the kubernetes pod. We noticed that the guaranteed throughput provided by iperf analysis could not exceed 100 mbps between the different equipments and the jitter was highly increased reaching over 4 ms in some cases. This high QoS degradation can be explained by the important number of packet's encapsulation during transmission engendering IP header overhead, where packets go through several layers to attend their destinations. To overcome this QoS deterioration, we suggested installing the `Kuryr`[3] which facilitated the communication between Kubernetes platform and openstack based platform, `OpenCif`. Moreover, to improve the network management, we integrated the `ONOS` SDN controller by replacing the Neutron module. Then, by adding `ONOS`, we could deploy optimized routing with an enhanced data plane layer. The obtained results were highly improved and we reached a guaranteed throughput equals to 1 Gbps between a selected VM and a pod installed in different compute nodes. This results can be explained by the fact that the communication throughput was limited by the NIC physical server interface which is equal to 1 Ethernet Gbps. However, for intra-compute communication, we noticed that the throughput could reach between a minimum of 6.4 Gbps to 7.1 Gbps. In fact, this variation depended on the computing and storage resources that exist in the hosting server. On the other hand, the jitter was deeply minimized to reach about 0.5 ms when the VM and Pod are communicating in the same compute node. Finally, we conclude that the proposed architecture has improved the QoS performance in comparison with a traditional synchronization between OpenStack and Kubernetes platforms. The project was carried out as a part of a graduation project for cloud and DevOps interns.

9. ***Level 3 Support Manager of Devocloud support and maintenance*** *2017-2019*

   Once the `Devocloud` platform of DRI was operational, I implemented a support process where we provided to our clients the ability to have virtual machines on-demand in order to develop their research projects and to improve their skills by testing new technologies. The support platform was based on GLPI IT Asset management and the team was composed of system and network trainees. They were responsible for the aforementioned

[3] `Kuryr` is an Openstack project permitting the deployment of native network based on Neutron in Kubernetes.

platform maintenance in order to build the latest OpenStack updates and to ensure a stable service to our clients. They also were responsible for treating and resolving the clients' tickets. Whereas, for more complicated client's demands or a specified technical issue, I was involved to find the best solution that fits both client needs and DRI resources availability. Besides, I was responsible for the whole DRI IT design and maintenance i.e., optimizing the location of the requested VMs in the different managed platforms. It is noteworthy to mention that my principle mission was to ensure the learning and skills improvement of the involved trainees.

10. ***DRI services evolution*** *2017-2019*

    In the first step of the project, I worked on the enhancement of DRI backup by implementing and deploying a newly designed storage platform in order to optimize the VMs access and archive DRI data through a specified classification according to the users' profiles and access rights. Also, I managed the migration of `Kaizen`[4] from a physical platform to the DRI private cloud in order to provide a flexible and easy access to the consultants from all Devoteam locations in France. Moreover, I proposed to install a DRI GitLab for our clients i.e., consultants based in DRI in order to submit their codes and project's reports in a centralized dedicated platform. In addition to that, I participated in developing a new DRI website presenting several DRI activities and services. It is noteworthy to say that I also relied on two system and network administrators to accomplish the aforementioned missions.

My mission for the overall projects basically consists on defining the projects topics, managing the team members, supervising the interns and ensuring the integration of the projects results in an environment of production. Note that the duration of the projects is varying between 6 months and 1 year each. Besides, considering the formulation of the project topics, I get inspired from my different contributions in my thesis and from the evolution of the different technologies emerging in the market. By doing so, my objective was not only to test their feasibility in a real environment but also to propose products that can fit our clients needs e.g., the emerging SDN controllers (such as ONOS after Opendaylight) and/or DevOps technologies (such as Kubernetes, Ansible, Teraform, etc.). My goals were i) to make in practise the different skills and knowledge earned in this thesis within a real production environment, ii) to propose different projects that can enrich the thesis fields by including new research areas, and iii) to suggest some proof of concept of several cloud and/or SDN based environments that can fulfill our clients needs. Furthermore, managing these projects allowed me to identify the difference between academic and industrial environments. Moreover, this experience was an occasion to improve both my technical and managerial skills. Indeed, during this period, I passed certification exams related to management skills. So, I got certified by Prince 2 certification, ITIL v3 Foundation certification, Time management Certification and I got a Process Com training to learn how to manage and communicate with team members despite their different profiles and skills.

[4]Kaizen is a web platform that helps DRI managers to have information about consultants participation and /or presence in their projects.

List of figures

List of tables

Bibliography

[1] "At Ignite, Microsoft turns the spotlight on productivity with Project Cortex and Fluid," 2019. [Online]. Available: https://siliconangle.com/2019/11/04/microsoft-turns-spotlight-productivity-ignite-project-cortex-fluid/

[2] "The evolution of the data center," 2014. [Online]. Available: https://siliconangle.com/2014/03/05/the-evolution-of-the-data-center-timeline-from-the-mainframe-to-the-cloud-tc0114/

[3] " What is a Bare Metal Hypervisor VMware Glossary." [Online]. Available: https://www.vmware.com/topics/glossary/content/bare-metal-hypervisor

[4] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, 2010.

[5] P. Mell, T. Grance, *et al.*, "The NIST definition of cloud computing," *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology (NIST)*, 2011.

[6] M. E. I. Malaainine, H. Rhinane, L. Baidder, and H. Lechger, "Omt-g modeling and cloud implementation of a reference database of addressing in Morocco," *Journal of Geographic Information System*, vol. 5, no. 3, p. 235, 2013.

[7] "By 2019, 60% of IT workloads will run in the cloud," 2017. [Online]. Available: https://451research.com/blog/1910-by-2019,-60-of-it-workloads-will-run-in-the-cloud

[8] "Global IT workload distribution by cloud type 2020," 2021. [Online]. Available: https://www.statista.com/statistics/748238/worldwide-it-cloud-workload-distribution/

[9] "Canalys Newsroom- Canalys: Cloud spend to surpass US$143 billion in 2020, driven by IT channel," 2019. [Online]. Available: https://www.canalys.com/newsroom/canalys-cloud-spend-to-surpass-us143-billion-in-2020-driven-by-it-channel

[10] "Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17% in 2020," 2019. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-11-13-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2020

[11] "Cloud Giants Continue Pouring Billions Into Data Centers," 2018. [Online]. Available: https://www.datacenterknowledge.com/cloud/cloud-giants-continue-pouring-billions-data-centers

[12] "Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper," 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[13] P. Wang, R. X. Gao, and Z. Fan, "Cloud Computing for Cloud Manufacturing: Benefits and Limitations," *Journal of Manufacturing Science and Engineering*, vol. 137, no. 4, 2015.

[14] 5G Infrastructure Association and others, "The 5G infrastructure public private partnership: The next generation of communication networks and services," 2015.

[15] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI white paper*, 2015.

[16] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on Mobile Edge Computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[17] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile Edge Computing: A taxonomy," *Proceeding of the Sixth International Conference on Advances in Future Internet*, 2014.

[18] P. Mach and Z. Becvar, "Mobile Edge Computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[19] J. Lee and J. Lee, "Hierarchical Mobile Edge Computing Architecture based on context awareness," *Applied Sciences*, vol. 8, no. 7, p. 1160, 2018.

[20] D. Abts and B. Felderman, "A guided tour of data-center networking," *Communications of the ACM*, vol. 55, no. 6, p. 44, 2012.

[21] "Data centers: 2018 year in review," 2019. [Online]. Available: https://engineering.fb.com/data-center-engineering/data-centers-2018

[22] N. Jones, "How to stop data centres from gobbling up the world's electricity," *Nature*, 2018. [Online]. Available: https://www.nature.com/articles/d41586-018-06610-y

[23] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 640–656, 2017.

[24] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "A comparative analysis of data center network architectures," *IEEE International Conference on Communications (ICC)*, 2014.

[25] J. Kim, W. Dally, J. Dally, and D. Abts, "Adaptive Routing in High-Radix Clos Network," *ACM/IEEE Conference on Supercomputing*, 2006.

[26] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2008.

[27] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A Scalable Fault-tolerant Layer 2 Data Center Network Fabric," *ACM SIGCOMM computer communication review*, vol. 39, no. 4, pp. 39–50, 2009.

[28] 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," *ACM SIGCOMM computer communication review*, vol. 39, no. 4, pp. 51–62, 2009.

[29] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the Social Network's (Datacenter) Network," *Proceedings of the ACM Conference on Special Interest Group on Data Communication*, 2015.

[30] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," *Proceedings of the ACM Conference on Special Interest Group on Data Communication*, 2015.

[31] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 498–511, 2013.

[32] L. Peng, C.-H. Youn, W. Tang, and C. Qiao, "A Novel Approach to Optical Switching for Intradatacenter Networking," *Journal of Lightwave Technology*, vol. 30, no. 2, pp. 252–266, 2012.

[33] S. Talebi, F. Alam, I. Katib, M. Khamis, R. Salama, and G. N. Rouskas, "Spectrum management techniques for elastic optical networks: A survey," *Optical Switching and Networking*, 2014.

[34] X. Ye, Y. Yin, S. Yoo, P. Mejia, R. Proietti, and V. Akella, "DOS - A scalable optical switch for datacenters," *ACM / IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2010.

[35] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers," *Proceedings of the ACM SIGCOMM Conference*, 2010.

[36] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos, "A survey on data center networking for cloud computing," *Computer Networks*, vol. 91, pp. 528–547, 2015.

[37] Y. Cui, H. Wang, X. Cheng, and B. Chen, "Wireless data center networking," *IEEE Wireless Communications*, vol. 18, no. 6, pp. 46–53, 2011.

[38] J.-Y. Shin, E. G. Sirer, H. Weatherspoon, and D. Kirovski, "On the Feasibility of Completely Wirelesss Datacenters," *IEEE / ACM Transactions on Networking*, vol. 21, no. 5, pp. 1666–1679, 2013.

[39] B. Dab, I. Fajjari, and N. Aitsaadi, "A novel joint routing and channel allocation approach in hybrid data center network," *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2017.

[40] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, p. 75, 2008.

[41] 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," *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2009.

[42] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: Using Backup Port for Server Interconnection in Data Centers," *IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[43] D. S. Marcon, R. R. Oliveira, L. P. Gaspary, and M. P. Barcellos, "Chapter 4 Data Center Networks And Relevant Standards," 2015.

[44] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly, "Symbiotic Routing in Future Data Centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 51–62, 2011.

[45] P. Costa, A. Donnelly, G. O'Shea, and A. Rowstron, "CamCubeOS: A Key-based Network Stack for 3d Torus Cluster Topologies," *ACM International Symposium on High-performance Parallel and Distributed Computing(HPDC)*, 2013.

[46] "To Get to Six Data Centers Per Region, Facebook Rethinks Its Network (Again)," 2019. [Online]. Available: https://www.datacenterknowledge.com/facebook/get-six-data-centers-region-facebook-rethinks-its-network-again

[47] "Facebook to Expand Prineville Data Center Facebook," 2010. [Online]. Available: https://www.facebook.com/notes/prineville-data-center/facebook-to-expand-prineville-data-center/411605058132/

[48] "Facebook begins expansion of data center in Odense," 2020. [Online]. Available: https://www.datacenter-forum.com/datacenter-forum/facebook-begins-expansion-of-data-center-in-odense

[49] H. Qi, M. Shiraz, J.-y. Liu, A. Gani, Z. Abdul Rahman, and T. A. Altameem, "Data center network architecture in cloud computing: review, taxonomy, and open research issues," *Journal of Zhejiang University SCIENCE C*, vol. 15, no. 9, pp. 776–793, 2014.

[50] K. Kant, "Data center evolution: A tutorial on state of the art, issues, and challenges," *Computer Networks*, 2009.

[51] A. R. Curtis, T. Carpenter, M. Elsheikh, A. López-Ortiz, and S. Keshav, "Rewire: An optimization-based framework for unstructured data center network design," *IEEE Conference on Computer Communications (INFOCOM)*, 2012.

[52] A. R. Curtis, S. Keshav, and A. Lopez-Ortiz, "LEGUP: Using heterogeneity to reduce the cost of data center network upgrades," *Proceedings of the 6th International Conference*, 2010.

[53] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[54] A. Singla, P. B. Godfrey, and A. Kolla, "High throughput data center topology design," *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2014.

[55] "Bisectional Bandwidth. And why L2MP and Trill/RBridges is important ?" 2010. [Online]. Available: https://etherealmind.com/bisectional-bandwidth-l2mp-trill-bridges-design-value/

[56] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[57] Y. Liu and J. Muppala, "Fault-tolerance characteristics of data center network topologies using fault regions," *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2013.

[58] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2008.

[59] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast congestion control for TCP in datacenter networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 21, no. 2, pp. 345–358, 2013.

[60] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, "Data center transport mechanisms: Congestion control theory and IEEE standardization," *46th Annual Allerton Conference on Communication, Control, and Computing*, 2008.

[61] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar, "Af-QCN: Approximate fairness with quantized congestion notification for multi-tenanted data centers," *18th IEEE symposium on high performance interconnects*, 2010.

[62] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella, "AC/DC TCP: Virtual congestion control enforcement for datacenter networks," *Proceedings of the ACM SIGCOMM Conference*, 2016.

[63] K. Xi, Y. Liu, and H. J. Chao, "Enabling flow-based routing control in data center networks using probe and ECMP," *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2011.

[64] S. Radhakrishnan, M. Tewari, R. Kapoor, G. Porter, and A. Vahdat, "Dahu: Commodity switches for direct connect data center networks," *Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems*, 2013.

[65] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. V. Vasilakos, "Survey on routing in data centers: insights and future directions," *IEEE network*, vol. 25, no. 4, pp. 6–10, 2011.

[66] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," *ACM Proceedings of the 6th International Conference*, 2010.

[67] Z. Guo, J. Duan, and Y. Yang, "On-line multicast scheduling with bounded congestion in fat-tree data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 102–115, 2013.

[68] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, *et al.*, "Hedera: dynamic flow scheduling for data center networks." *NSDI*, 2010.

[69] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 779–792, 2016.

[70] H. Yao and W. Muqing, "SDN-based routing mechanism for cloud data centers," *IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2017.

[71] "Wiki HomeONOSWiki." [Online]. Available: https://wiki.onosproject.org/

[72] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," *7th conference on Networked systems design and implementation (USENIX)*, vol. 10, pp. 249–264, 2010.

[73] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a next generation data center architecture: scalability and commoditization," *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, 2008.

[74] R. Zhang-Shen, "Valiant load-balancing: Building networks that can support all traffic matrices," *Springer Algorithms for next generation networks*, 2010.

[75] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 338–347, 2010.

[76] M. Csernai, F. Ciucu, R.-P. Braun, and A. Gulyás, "Reducing cabling complexity in large flattened butterfly networks by an order of magnitude," *Optical Fiber Communication Conference*, 2014.

[77] M. Csernai, F. Ciucu, R.-P. Braun, and A. Gulyás, "Towards 48-fold cabling complexity reduction in large flattened butterfly networks," *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[78] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 126–137, 2007.

[79] D. Lin, Y. Liu, M. Hamdi, and J. Muppala, "Flatnet: Towards a flatter data center network," *IEEE Global Communications Conference (GLOBECOM)*, 2012.

[80] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A cost comparison of datacenter network architectures," *Proceedings of the 6th International Conference*, 2010.

[81] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: a high performance network structure for modular data center interconnection," *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009.

[82] A. Vahdat, H. Liu, X. Zhao, and C. Johnson, "The emerging optical data center," *Optical Fiber Communication Conference*, 2011.

[83] N. Farrington, G. Porter, P.-C. Sun, A. Forencich, J. Ford, Y. Fainman, G. Papen, and A. Vahdat, "A demonstration of ultra-low-latency data center optical circuit switching," *ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, 2012.

[84] Perelló, Jordi and Spadaro, Salvatore and Ricciardi, Sergio and Careglio, Davide and Peng, Shuping and Nejabati, Reza and Zervas, George and Simeonidou, Dimitra and Predieri, Alessandro and Biancani, Matteo and others, "All-optical packet/circuit switching-based data center network for enhanced scalability, latency, and throughput," *IEEE Network*, vol. 27, no. 6, pp. 14–22, 2013.

[85] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 498–511, 2013.

[86] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 339–350, 2011.

[87] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-Through: Part-time optics in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 327–338, 2011.

[88] D. Chan and D. R. Oran, "De-congesting data centers with wireless point-to-multipoint flyways," 2017.

[89] S. Kandula, J. Padhye, and V. Bahl, "Flyways to de-congest data center networks," 2009.

[90] Y. Cui, H. Wang, X. Cheng, and B. Chen, "Wireless data center networking," *IEEE Wireless Communications*, vol. 18, no. 6, pp. 46–53, 2011.

[91] Y. Zhu, X. Zhou, Z. Zhang, L. Zhou, A. Vahdat, B. Y. Zhao, and H. Zheng, "Cutting the cord: A robust wireless facilities network for data centers," *Proceedings of the 20th annual international conference on Mobile computing and networking*, 2014.

[92] W. Zhang, X. Zhou, L. Yang, Z. Zhang, B. Y. Zhao, and H. Zheng, "3D beamforming for wireless data centers," *Proceedings of the 10th ACM workshop on hot topics in networks*, 2011.

[93] B. Elspas, W. H. Kautz, and J. Turner, "Ttheory of cellular logic networks and machines," STANFORD RESEARCH INST MENLO PARK CALIF, Tech. Rep., 1968.

[94] K. Han, Z. Hu, J. Luo, and L. Xiang, "RUSH: Routing and scheduling for hybrid data center networks," *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[95] J.-Y. Shin, E. G. Sirer, H. Weatherspoon, and D. Kirovski, "On the feasibility of completely wirelesss datacenters," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1666–1679, 2013.

[96] N. Alon and Y. Roichman, "Random Cayley graphs and expanders," *Random Structures & Algorithms*, vol. 5, no. 2, pp. 271–284, 1994.

[97] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: A survey," *Journal of Parallel and Distributed Computing*, vol. 96, pp. 45–74, 2016.

[98] L. Gyarmati and T. A. Trinh, "Scafida: A scale-free network inspired data center architecture," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 5, pp. 4–12, 2010.

[99] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[100] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: A survey," *Journal of Parallel and Distributed Computing*, vol. 96, pp. 45–74, 2016.

[101] L. Strigini, "Fault tolerance and resilience: meanings, measures and assessment," *Resilience assessment and evaluation of computing systems*, 2012.

[102] J. Kozlowicz, "High Availability vs. Fault Tolerance vs. Disaster Recovery | Green House Data," 2018. [Online]. Available: https://www.greenhousedata.com/blog/high-availability-vs-fault-tolerance-vs-disaster-recovery

[103] W. Wojdak, "Rapid Spanning Tree Protocol: A new solution from an old technology," *Reprinted from CompactPCI Systems*, 2003.

[104] A. Iselt, A. Kirstadter, A. Pardigon, and T. Schwabe, "Resilient routing using MPLS and ECMP," *IEEE Workshop on High Performance Switching and Routing (HPSR)*, 2004.

[105] M. K. Khattak, Y. Tang, H. Fahim, E. Rehman, and M. F. Majeed, "Effective Routing Technique: Augmenting Data Center Switch Fabric Performance," *IEEE Access*, vol. 8, pp. 37 372–37 382, 2020.

[106] M. Noormohammadpour and C. S. Raghavendra, "Datacenter Traffic Control: Understanding Techniques and Tradeoffs," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1492–1525, 2017.

[107] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," *Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies*, 2011.

[108] X. Wang, J.-X. Fan, C.-K. Lin, J.-Y. Zhou, and Z. Liu, "BCDC: a high-performance, server-centric data center network," *Springer Journal of Computer Science and Technology*, vol. 33, no. 2, pp. 400–416, 2018.

[109] Z. Chkirbene, R. Hadjidj, S. Foufou, and R. Hamila, "LaScaDa: A Novel Scalable Topology for Data Center Network," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2051–2064, 2020.

[110] S. Kan, J. Fan, B. Cheng, and X. Wang, "The Communication Performance of BCDC Data Center Network," *IEEE 12th International Conference on Communication Software and Networks (ICCSN)*, 2020.

[111] Y. Yu and C. Qian, "Space Shuffle: A Scalable, Flexible, and High-Bandwidth Data Center Network," *IEEE 22Nd International Conference on Network Protocols (ICNP)*, 2014.

[112] P. Costa, A. Donnelly, G. O'Shea, and A. Rowstron, "Camcubeos: a key-based network stack for 3d torus cluster topologies," *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, 2013.

[113] X. Hong, Y. Yang, Y. Gong, and J. Chen, "Passive optical interconnects based on cascading wavelength routing devices for datacenters: A cross-layer perspective," *Journal of Optical Communications and Networking*, vol. 9, no. 4, pp. C45–C53, 2017.

[114] T. Gao, X. Li, B. Guo, S. Yin, W. Li, and S. Huang, "Spectrum-efficient multipath provisioning with content connectivity for the survivability of elastic optical data center networks," *Optical Fiber Technology*, vol. 36, pp. 353–365, 2017.

[115] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Proactive retention-aware caching with multi-path routing for wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1286–1299, 2018.

[116] J. Lacan and E. Lochin, "XOR-based Source Routing," *IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, 2020.

[117] D. Li, J. Yu, J. Yu, and J. Wu, "Exploring efficient and scalable multicast routing in future data center networks," *IEEE International Conference on Computer Communications(INFOCOM)*, 2011.

[118] A. Latif, P. Kathail, S. Vishwarupe, S. Dhesikan, and A. Khreishah, "IRP: Intelligent rendezvous point for multicast control plane," *IEEE 38th Sarnoff Symposium*, 2017.

[119] A. Latif, P. Kathail, S. Vishwarupe, S. Dhesikan, A. Khreishah, and Y. Jararweh, "Multicast Optimization for CLOS Fabric in Media Data Centers," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1855–1868, 2019.

[120] A. Latif, R. Paul, R. Chhibber, A. Singh, R. Parameswaran, A. Khreishah, and Y. Jararweh, "DiRP: Distributed Intelligent Rendezvous Point for Multicast Control Plane," *Ninth International Green and Sustainable Computing Conference (IGSC)*, 2018.

[121] G. N. Rouskas and I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," *IEEE Conference on Computer Communications*, vol. 1, pp. 353–360, 1996.

[122] S. Luo, H. Xing, and K. Li, "Near-Optimal Multicast Tree Construction in Leaf-Spine Data Center Networks," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2581–2584, 2020.

[123] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms (3-rd edition)," *MIT Press and McGraw-Hill*, 2009.

[124] S. Misra, A. Mondal, and S. Khajjayam, "Dynamic Big-Data Broadcast in Fat-Tree Data Center Networks With Mobile IoT Devices," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2898–2905, 2019.

[125] C. Wu, C. Ku, J. Ho, and M. Chen, "A Novel Pipeline Approach for Efficient Big Data Broadcasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 17–28, 2016.

[126] Z. Guo and Y. Yang, "Multicast fat-tree data center networks with bounded link oversubscription," *IEEE International Conference on computer Communications (INFOCOM)*, 2013.

[127] D. Lopez-Pajares, J. Alvarez-Horcajo, E. Rojas, J. A. Carral, and G. Ibanez, "Iterative Discovery of Multiple Disjoint Paths in Switched Networks with Multicast Frames," *IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018.

[128] Y. Guo, F. Kuipers, and P. Van Mieghem, "Link-disjoint paths for reliable QoS routing," *International Journal of Communication Systems*, vol. 16, no. 9, pp. 779–798, 2003.

[129] X. Gao, T. Chen, Z. Chen, and G. Chen, "NEMO: novel and efficient multicast routing schemes for hybrid data center networks," *Elsevier Computer Networks*, vol. 138, pp. 149–163, 2018.

[130] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "ESM: Efficient and scalable data center multicast routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 944–955, 2011.

[131] X. Li, L. Zhang, Y. Tang, J. Guo, and S. Huang, "Distributed sub-tree-based optical multicasting scheme in elastic optical data center networks," *IEEE Access*, vol. 6, pp. 6464–6477, 2018.

[132] T. Gao, W. Zou, X. Li, B. Guo, S. Huang, and B. Mukherjee, "Distributed sub-light-tree based multicast provisioning with shared protection in elastic optical datacenter networks," *Elsevier Optical Switching and Networking*, vol. 31, pp. 39–51, 2019.

[133] C. Garrido, A. Leiva, and A. Beghelli, "A rmlsa algorithm with modulation format conversion at intermediate nodes," *19th International Conference on Transparent Optical Networks (ICTON)*, 2017.

[134] M. Moharrami, A. Fallahpour, H. Beyranvand, and J. A. Salehi, "Resource allocation and multicast routing in elastic optical networks," *IEEE Transactions on Communications*, vol. 65, no. 5, pp. 2101–2113, 2017.

[135] H. Rastegarfar, L. Yan, K. Szczerba, and E. Agrell, "Pam performance analysis in multicast-enabled wavelength-routing data centers," *Journal of Lightwave Technology*, vol. 35, no. 13, pp. 2569–2579, 2017.

[136] K. Zheng, Y. Bai, and X. Wang, "FCTcon: Dynamic Control of Flow Completion Time in Data Center Networks for Power Efficiency," *IEEE Transactions on Cloud Computing*, 2019.

[137] J. Huang, T. He, Y. Huang, and J. Wang, "ARS: Cross-layer adaptive request scheduling to mitigate TCP incast in data center networks," *IEEE 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2016.

[138] D. Shan and F. Ren, "Improving ecn marking scheme with micro-burst traffic in data center networks," *IEEE Conference on Computer Communications (INFOCOM)*, 2017.

[139] S. Maroulis, N. Zacheilas, and V. Kalogeraki, "A holistic energy-efficient real-time scheduler for mixed stream and batch processing workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2624–2635, 2019.

[140] L. Luo, D. Guo, J. Wu, T. Qu, T. Chen, and X. Luo, "VLCcube: A VLC Enabled Hybrid Network Structure for Data Centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2088–2102, 2016.

[141] Y. Qin, D. Guo, G. Tang, and B. Ren, "TIO: A VLC-enabled hybrid data center network architecture," *Tsinghua Science and Technology*, vol. 24, no. 4, pp. 484–496, 2019.

[142] B. Dab, I. Fajjari, and N. Aitsaadi, "Online-batch joint routing and channel allocation for hybrid data center networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 831–846, 2017.

[143] B. Dab, I. Fajjari, and N. Aitsaadi, "A Heuristic Approach for Joint Batch-Routing and Channel Assignment in Hybrid-DCNs," *IEEE Global Communications Conference (GLOBECOM)*, 2017.

[144] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2014.

[145] Z. Jia, Y. Sun, Q. Liu, S. Dai, and C. Liu, "cRetor: An SDN-Based Routing Scheme for Data Centers With Regular Topologies," *IEEE Access*, vol. 8, pp. 116 866–116 880, 2020.

[146] F. Rhamdani, N. A. Suwastika, and M. A. Nugroho, "Equal-cost multipath routing in data center network based on software defined network," *6th International Conference on Information and Communication Technology (ICoICT)*, 2018.

[147] H. Yao and W. Muqing, "SDN-based routing mechanism for cloud data centers," *IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2017.

[148] Y.-C. Wang and S.-Y. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1422–1434, 2018.

[149] S. Luo, H. Yu, K. Li, and H. Xing, "Efficient file dissemination in data center networks with priority-based adaptive multicast," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1161–1175, 2020.

[150] Q. Fu, E. Sun, K. Meng, M. Li, and Y. Zhang, "Deep Q-Learning for Routing Schemes in SDN-Based Data Center Networks," *IEEE Access*, vol. 8, pp. 103 491–103 499, 2020.

[151] J. Ho, D. W. Engels, and S. E. Sarma, "HiQ: a hierarchical Q-learning algorithm to solve the reader collision problem," *International Symposium on Applications and the Internet Workshops (SAINTW'06)*, 2006.

[152] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2014.

[153] A. Shirmarz and A. Ghaffari, "Performance issues and solutions in SDN-based data center: A survey," *Springer, The Journal of Supercomputing*, vol. 76, no. 10, pp. 7545–7593, 2020.

[154] C.-C. Chuang, Y.-J. Yu, and A.-C. Pang, "Flow-aware routing and forwarding for sdn scalability in wireless data centers," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1676–1691, 2018.

[155] M.-H. Tsai, C.-C. Chuang, S.-F. Chou, A.-C. Pang, and G.-Y. Chen, "Enabling Efficient and Consistent Network Update in Wireless Data Centers," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 505–520, 2019.

[156] M. Yang, H. Rastegarfar, and I. B. Djordjevic, "Physical-layer adaptive resource allocation in software-defined data center networks," *Journal of Optical Communications and Networking*, vol. 10, no. 12, pp. 1015–1026, 2018.

[157] N. Panahi, D. Careglio, and J. S. Pareta, "A Hybrid Packet/Circuit Optical Transport Architecture for DCN," *21st International Conference on Transparent Optical Networks (ICTON)*, 2019.

[158] M. Zeng, Y. Li, W. Fang, W. Lu, X. Liu, H. Yu, and Z. Zhu, "Control plane innovations to realize dynamic formulation of multicast sessions in inter-dc software-defined elastic optical networks," *Optical Switching and Networking*, vol. 23, pp. 259–269, 2017.

[159] A. Morea, "SMART-A: An SDN Application for Reconfiguring Optical Networks," *IEEE Italian Conference on Optics and Photonics (ICOP)*, 2020.

[160] S. Shukla, P. Ranjan, and K. Singh, "MCDC: Multicast routing leveraging SDN for Data Center networks," *International Conference - Cloud System and Big Data Engineering (CONFLUENCE)*, 2016.

[161] J. Qadir, N. Ahad, E. Mushtaq, and M. Bilal, "SDNs, clouds, and big data: new opportunities," *12th International Conference on Frontiers of Information Technology*, 2014.

[162] M. Paliwal and D. Shrimankar, "Effective Resource Management in SDN Enabled Data Center Network Based on Traffic Demand," *IEEE Access*, vol. 7, pp. 69 698–69 706, 2019.

[163] SDN_Paris, "Onos overview meetup sdn paris - redux," 2016. [Online]. Available: https://www.slideshare.net/SDNParis/onos-overview-meetup-sdn-paris-redux

[164] "System Components - ONOS - Wiki," 2016. [Online]. Available: https://wiki.onosproject.org/ display/ONOS/System&+Components

[165] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2014.

[166] S. Asadollahi, B. Goswami, and M. Sameer, "Ryu controller's scalability experiment on software defined networks," *IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, 2018.

[167] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several software defined networking controllers," *IEEE International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, 2015.

[168] S. Secci, K. Attou, D. Phung, S. Scott-Hayward, D. Smyth, S. Vemuri, and Y. Wang, "Onos security and performance analysis: Report no. 1," 2017.

[169] S. Secci, A. Diamanti, J. Sanchez, M. Vilchez, M. T. Bah, P. Vizarreta, C. Machuca, S. Scott-Hayward, and D. Smith, "Security and performance comparison of onos and odl controllers," 2019.

[170] "Security & performance analysis brigade - ONOS - Wiki," 2021. [Online]. Available: https://wiki.onosproject.org/pages/viewpage.action?pageId=12422167

[171] "Onos overview meetup sdn paris - redux," 2016, library Catalog: SlideShare. [Online]. Available: https://www. slideshare.net/SDNParis/onos-overview-meetup-sdn-paris-redux

[172] Open Networking Foundation, "OpenFlow Switch Specification - Version 1.4.1 (Protocol version 0x05)," 2015. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.4.1.pdf

[173] "Mininet Overview - Mininet." [Online]. Available: http://mininet.org/overview/

[174] "Mininet Python API Reference Manual: Class List." [Online]. Available: http://mininet.org/api/annotated.html

[175] "onosproject/onos-Docker-Hub." [Online]. Available: https://hub.docker.com/r/onosproject/ onos/

[176] S. Secci, S. Scott-Hayward, Y. Wang, Q. P. Van, and A. Campanella, "ONOS Security and Performance Analysis (Report No. 2)," 2018.

[177] "Welcome to Python.org." [Online]. Available: https://www.python.org/

[178] "Setting up the Python API of CPLEX." [Online]. Available: www.ibm.com/support/knowledgecenter/sssa5p_12.7.1/ilog.odms.cplex.help/cplex/ gettingstarted/topics/set_up/python_setup.html

[179] S. Jose, "QoS: Congestion Management Configuration Guide, Cisco IOS XE Release 3S - Configurable Queue Depth," 2016.

[180] "MaximumTransmissionUnit - eduPERT KB - GÉANT federated confluence," 2018. [Online]. Available: https://wiki.geant.org/display/public/EK/MaximumTransmissionUnit

[181] D. G. Dutt, *Cloud Native Data Center Networking*. O'Reilly Media, Inc., 2019.

[182] "Connecting Servers to Leaf Switches - Huawei DCN Design Guide - Huawei." [Online]. Available: https://support.huawei.com/enterprise/en/doc/Name=connecting-servers-to-leaf-switches

[183] A. Singla and B. Rijsman, "Day One: Understanding OpenContrail Architecture," 2013.

[184] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Performance of Network Virtualization in cloud computing infrastructures: The OpenStack case," *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, 2014.

[185] "Ant colony optimization algorithms." [Online]. Available: https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithm

[186] M. Dorigo, "The Ant Colony Optimization Metaheuristic: Algorithms, Applications and Advances," *Springer, In Handbook of Metaheuristics*, 2002.

[187] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European journal of operational research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[188] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," *Springer, Handbook of metaheuristics*, 2019.

[189] I. Fajjari, N. Aitsaadi, M. Pióro, and G. Pujolle, "A new virtual network static embedding strategy within the Cloud's private backbone network," *Elsevier Computer Networks*, vol. 62, pp. 69–88, 2014.

[190] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[191] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *USENIX on Hot topics in Cloud Computing HotCloud*, vol. 10, no. 10, p. 95, 2010.

[192] H. Wang, H. Xu, S. Yi, and Z. Shi, "A tree-growth based ant colony algorithm for QoS multicast routing problem," *Elsevier Expert Systems with Applications*, vol. 38, no. 9, pp. 11 787–11 795, 2011.

[193] M. K. Patel, M. R. Kabat, and C. R. Tripathy, "A hybrid ACO/PSO based algorithm for QoS multicast routing problem," *Elsevier Ain Shams Engineering Journal*, vol. 5, no. 1, pp. 113–120, 2014.

[194] K. Deierling, "In Modern Datacenters,The Latency Tail Wags The Network Dog," 2018. [Online]. Available: https://www.nextplatform.com/2018/03/27/in-modern-datacenters-the-latency-tail-wags-the-network-dog/

[195] W. R. Stevens, M. Allman, and V. Paxson, "TCP Congestion Control," 1999. [Online]. Available: https://tools.ietf.org/html/rfc2581

[196] "Unicast, Broadcast, and Multicast," 2020. [Online]. Available: https://erg.abdn.ac.uk/users/gorry/eg3567/intro-pages/uni-b-mcast.html

[197] A. C. Caputo, "Digital video surveillance and security," 2014.

[198] Y. Tanigawa, Y. Umeno, and H. Tode, "Packet Transmission Scheduling for Enhancing Power Saving and TCP Throughput Performance in Wireless LAN with Multicast/Unicast Flows," *IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2018.

[199] T. Zhu, D. Feng, F. Wang, Y. Hua, Q. Shi, Y. Xie, and Y. Wan, "A congestion-aware and robust multicast protocol in SDN-based data center networks," *Journal of Network and Computer Applications*, vol. 95, pp. 105–117, 2017.

[200] "iPerf - iPerf3 and iPerf2 user documentation." [Online]. Available: https://iperf.fr/iperf-doc.php#multicast

[201] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, pp. 65–77, 2002.

[202] O. Soualah, I. Fajjari, N. Aitsaadi, and A. Mellouk, "A batch approach for a survivable virtual network embedding based on monte-carlo tree search," *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[203] F. Zegrari, A. Idrissi, and H. Rehioui, "Resource allocation with efficient load balancing in cloud environment," *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, 2016.

[204] E. Altman, K. Avrachenkov, and A. Garnaev, "Generalized α-fair resource allocation in wireless networks," *47th IEEE Conference on Decision and Control*, 2008.

[205] A. Jean-Marie and A. Dandoush, "Flow-Level Modeling of Parallel Download in Distributed Systems," *Third International Conference on Communication Theory, Reliability, and Quality of Service*, 2010.

[206] J. M. Wang, Y. Wang, X. Dai, and B. Bensaou, "SDN-based multi-class QoS-guaranteed inter-data center traffic management," *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, 2014.

[207] Y. Feng and B. Li and B. Li, "Postcard: Minimizing Costs on Inter-Datacenter Traffic with Store-and-Forward," *32nd International Conference on Distributed Computing Systems Workshops*, 2012.

[208] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[209] A. Taha, S. Manzoor, and N. Suri, "SLA-based service selection for multi-cloud environments," *IEEE International Conference on Edge Computing (EDGE)*, 2017.

[210] A. G. García, I. B. Espert, and V. H. García, "SLA-driven dynamic cloud resource management," *Future Generation Computer Systems*, vol. 31, pp. 1–11, 2014.

[211] S. Singh, I. Chana, and R. Buyya, "STAR: SLA-aware autonomic management of cloud resources," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1040–1053, 2020.

[212] R. B. Uriarte, R. De Nicola, V. Scoca, and F. Tiezzi, "Defining and guaranteeing dynamic service levels in clouds," *Elsevier Future Generation Computer Systems*, vol. 99, pp. 27–40, 2019.