A Novel Placement Algorithm for the Controllers Of the Virtual Networks (COVN) in SD-WAN with Multiple VNs

Submitted for the Degree of Doctor of Philosophy

At the University of Northampton

2018

Ameer Mosa Al-Sadi

DEDICATION

TO MY FATHER

Mosa Thoeny Al-Sadi

For his inspiration and sacrifices .Without his incredible support I could not have completed my studies.

TO MY MOTHER
Bushraa Sadiq Gaffory

For her ceaseless prayers, encouragement and endless love.

TO MY WIFE
Duaa Hiyder Gaffory

For her endless love and support.

TO MY Family And Friends

May GOD bless you.

DECLARATION

I hereby declare that the work described in this thesis is original work undertaken by me for the degree of Doctor of Philosophy, at the Department of Computer Science and Immersive Technologies – University of Northampton, United Kingdom. No part of the material described in this thesis has been submitted for any award of any other degree or qualification in this or any other University or College of advanced education.

Ameer Mosa Al-Sadi

ACKNOWLEDGMENTS

God almighty is worthy all the thankful for empowering me by the motivation, the ability and the individuals who support the long PhD journey.

First, this work would not have been possible without the financial support of the Iraqi Ministry of Higher Education and Scientific Research and the University of Technology in Baghdad.

Immeasurable appreciation and deepest gratitude to my supervisor Dr. Ali Al-Sherbaz for his incredible support and guidance, with keeping the independence of research directions and decisions, which improve my research skills and experience. I would also greatly appreciate his care and concern throughout my PhD study.

Besides my advisor, I would like to express the truly indebted and thankful to the members of my supervisory team Dr. James Xue and Dr. Scott Turner for their thoughtful and valuable advice, insightful suggestions in the research work and constant encouragement. Their feedback and constructive criticism have been a great asset to me.

I am also very grateful to all colleagues and the staff in the Department of Computer Science and Immersive Technology. Also, it is a great pleasure to thank my sincere friends Mr. Riyadh Abbas, Mr. Marwan Aldabbagh, Dr. Ahmad Al-Khalil and Miss. Leila Benseddik for their cooperative work and moral support.

I would like to extend my great thankful for my cousin Dr. Firas Al-saidi for his continues encouragement and supportive guidance through the PhD study.

I dedicate my thesis to my parents who their love, prayers, blessings, and inspiration is the true reason behind any success I have realised in my life.

Finally, I should always remember the support and endless wishes of my wife Duaa and my lovely children Anas and Aseel. I owe sincere and earnest thankfulness to my dearest Brothers Mr. Anor and Mr. Aeesr, and my sisters Miss. Anoar and Miss. Athar for their encouragement and support.

PUBLICATIONS

During the 4-years of the research study, the following papers were published within the Department of Computer Science and Immersive Technologies at The University of Northampton.

1. Al-Sadi, A., Al-Sherbaz, A., Xue, J. and Turner, S. J. (2018) **Developing an asynchronous technique to evaluate the performance of SDN HP Aruba switch and OVS**. In: *IEEE Computing Conference 2018.* London: IEEE.

2. Al- Al-Sadi, A., Al-Sherbaz, A., Xue, J. and Turner, S. J. (2016) **Routing algorithm optimization for Software Defined Network WAN**. In: *Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA) - IRAQ (9-10) May.* Baghdad, Iraq: **IEEE.**

3. Al-Sadi, A., Al-Sherbaz, A., Turner, S. J. and Xue, J. (2016) **The management of distributed software defined networks in smart cities**. Workshop presented to: *School of Science and Technology Annual Research Conference, Newton Building, The University of Northampton, 02 March 2016*.

4. Al-Sadi, A., Al-Sherbaz, A., Xue, J. and Turner, S. J. (2015) **The management of the future internet**. In: *8th Manchester Metropolitan University (MMU) Postgraduate Research Conference 2015: Innovation, Manchester Metropolitan University, 05 November 2015*.

5. Al-Sadi, A., Al-Sherbaz, A., Xue, J. and Turner, S. J. (2015) **The Management of the Future Internet.** In : Workshop presented to: *Annual Research Conference 2015 For Postgraduate Research Degree Students and Early Career Researchers, The University of Northampton, 17 June 2015*.

ABSTRACT

The escalation of communication demands and the emergence of new telecommunication concepts such as 5G cellular system and smart cities requires the consolidation of a flexible and manageable backbone network. These requirements motivated the researcher to come up with a new placement algorithm for the Controller of Virtual Network (COVN). This is because SDN and network virtualisation techniques (NFV and NV), are integrated to produce multiple virtual networks running on a single SD-WAN infrastructure, which serves the new backbone.

One of the significant challenges of SD-WAN is determining the number and the locations of its controllers to optimise the network latency and reliability. This problem is fairly investigated and solved by several controller placement algorithms where the focus is only on physical controllers.

The advent of the sliced SD-WAN produces a new challenge, which necessitates the SD-WAN controllers (physical controller/hosted server) to run multiple instances of controllers (virtual controllers). Every virtual network is managed by its virtual controllers. This calls for an algorithm to determine the number and the positions of physical and virtual controllers of the multiple virtual SD-WANs. According to the literature review and to the best of the author knowledge, this problem is neither examined nor yet solved. To address this issue, the researcher designed a novel COVN placement algorithm to compute the controller placement of the physical controllers, then calculate the controller placement of every virtual SD-WAN independently, taking into consideration the controller placement of other virtual SD-WANs.

COVN placement does not partition the SD-WAN when placing the physical controllers, unlike all previous placement algorithms. Instead, it identifies the nodes of the optimal reliability and latency to all switches of the network. Then, it partitions every VN separately to create its independent controller placement. COVN placement optimises the reliability and the latency according to the desired weights. It also maintains the load balancing and the optimal resources utilisation. Moreover, it supports the recovering of the controller failure.

This novel algorithm is intensively evaluated using the produced COVN simulator and the developed Mininet emulator. The results indicate that COVN placement achieves the required optimisations mentioned above. Also, the implementations disclose that COVN placement can compute the controller placement for a large network (754 switches) in very small computation time (49.53 s). In addition, COVN placement is compared to POCO algorithm. The outcome reveals that COVN placement provides better reliability in about 30.76% and a bit higher latency in about 1.38%. Further, it surpasses POCO by constructing the balanced clusters according to the switch loads and offering the more efficient placement to recover controller-failure.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

5G	5th generation mobile networks or 5th generation wireless systems
API	Application Programming Interface
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CAPEX	Capital Expenditure
CLI	Command Line Interface
COVN	Controllers Of the Virtual Networks
CSP	Communication Service Providers
ETSI	European Telecommunications Standards Institute
FCNC	Farthest-Cluster to Nearest-Controller
FIB	Forwarding Information Base
GUI	Graphical User Interface
HPE VAN	HPE Virtual Application Networks
ICN	Information Centric Networking
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
JVM	Java Virtual Machine
MAC	Media Access Control
NAT	Network Address Translation
NFV	Network Functions Virtualization
NOS	Network Operating System
NV	Network Functions Virtualization
ONF	Open Network Foundation
OPEX	Operating Expense
OS	Operating System
OVS	Open vSwitch
QoS	Quality of Service
REST	Representation State Transfer
RTT	Round Trip Time

SDN	Software-Defined Networking
SD-WAN	Software-Defined Wide Area Network
SFOP	Shortest-Feasible OpenFlow Path
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TE	Traffic Engineering
TLS	Transport Layer Security
UDP	User Datagram Protocol
VLAN-ID	Virtual Local Area Network-Identifier
VM	Virtual Machine
VPC	Virtual Privet Cloud
VPN	Virtual Private Network

CHAPTER ONE
INTRODUCTION

Today Internet is based on the developed model of the traditional network, which began in the early 1970s (Leiner *et al.*, 2009). It grew gradually to connect the whole world. Its services are always evolving to meet the technical needs of society. Previously, the Internet was used to communicate and transfer files between computers (Leiner *et al.*, 2009). Currently, it is used for almost every activity, which can be performed by people and machines. For instance, it displays video and audio, makes telephone calls, collects data and remotely controls devices. In the near future, it will be used as infrastructure for the Internet of Things (IoT), smart cities and 5G cellular technology, which serve a wider platform of application, such as automated cars and smart houses (Tadinada, 2014). Traffic engineering (TE) develops some protocols and mechanisms to optimise the performance of this network, however, it does not have any essential change on its design and structure (Awduche *et al.*, 2002). TE developments cannot fully satisfy the current and the approaching requirements of telecommunication because it cannot provide the dynamic utilisation of network resources. One case reveals that, the Communication Service Providers (CSP), presently leaves about 50 percent of network bandwidth unutilized due to its inability to use it dynamically (ACG RESEARCH, 2014). For example, CSP of North America reveals that they could provide a periodic bandwidth between data-centres for 1500 small to medium companies and 9000 enterprises if they can utilise the unused bandwidth (ACG RESEARCH, 2014). Another case discloses that, 57 network enterprises suffered a huge increment in the number of middleboxes, which approximately equals the number of routers in these networks. This because these enterprises increase the number of middleboxes in ordered to improve security, optimise the Quality of Service (QoS) and deliver new services, it is possible to avoid this problem if network devices could change their functionality (Kreutz *et al.*, 2015a).

The mentioned escalated of the communication demands, and the emergence of these new telecommunication concepts increases the needs for updating the abilities of the traditional networks (Loffreda, 2015). The forthcoming network requires updating its

topology and services dynamically using management software that facilitates and automates this operation. Also, it needs to run multiple logical networks over a single network infrastructure, similar to data centres nowadays. "The automation of dynamic behaviour is achieved by using a novel paradigm of the network that contains forwarding devices, which have central management, called Software-Define Network (SDN)" (Al-Sadi et al., 2016). SDN provides faster, cheaper and more flexible management for a network. In addition, it strongly supports Network Function Virtualization (NFV), which enables the network functions to be executed using the software components instead of physical devices (Al-Sadi et al., 2016). Therefore, NFV allows multiple logical networks to be run on the single physical network. Moreover, it allows faster deployment of new services or updates for the old ones.

Over the past few years, SDN has received great attention from the network industry and academia, for example, it is deployed by 72 percent of enterprises and 81 percent of universities in the US. However, it still in the establishing stage and the expected investment of this new-generation of the network will exceed 105 Billion Dollars per annum by 2020 (Cox *et al.*, 2017). This leads researchers to investigate the problems of SDN in order to solve them. One of the significant challenges of the SDN is called Controller Placement. SDN has a group of central management units called controllers, which instruct the network switches to forward the data flow from the sender to receiver. The controller placement is concerned with determining the optimal number and the distribution of these controllers over the network to optimise its latency and reliability (Heller *et al.*, 2012). The controller placement problem has been fairly investigated for SDN model, however, there is no single solution which could meet the requisites of all implementations.

Recently, the developers emphasised the necessity of combining the SDN with the NFV to create more flexible and powerful management and enable multi-tenancy over single SD-WAN (Fekih Ahmed *et al.*, 2015). The need for such a network to be the backbone of the forthcoming 5G network, is pushing them to combine SDN, NFV, and Network virtualisation (NV) into a 5G cellular system to design a complete framework of this model (Cox *et al.*, 2017). One example of this framework is the SoftAir (Akyildiz *et al.*, 2015). This evolution of the SDN framework should be addressed and reflected in the development of the controller placement algorithm to improve its optimisation.

1.1. Problem Description

With the emergence of SD-WAN, the research of controller placement algorithms has been arisen. This algorithm optimises the number and the location of the controllers as trade-off between or to combine the improvement of several objectives, such as latency, reliability load balancing and resources utilisation (Hock, Gebert, *et al.*, 2014). Lately, SD-WAN has been extended by the virtualisation technology to amplify its management flexibility and construct multiple VNs over its physical infrastructure (Zilong Ye et al., 2013). The evolution of SD-WAN from operating as a single physical slice to act as multiple virtual slices running over a single infrastructure, motivates the author to rethink the validity of the algorithms, which were developed recently to handle the controller placement problem of a single slice of SD-WAN. The author found that the existing placement algorithms cannot optimise the controller placement of the multiple virtual SD-WAN. The reason is that, in SD-WAN of the multiple VNs, every slice has the virtual equipment, which needs to swap dynamically and independent from the other slices. Therefore, its controller placement should be optimised, either independently or in some method that does not degrade the controller placement of other slices. Therefore, a novel algorithm of controller placement should be established for this new model of the multiple VNs.

The proposed novel algorithm to solve this challenge relates to the existing SDN controller placement algorithms (Zhang et al., 2011), but it has different requirements and solves the problem of the evolved SD-WAN model. The existing controller placement algorithms place the physical controllers in the single physical SDN network, whereas the proposed novel algorithm places a different type of controllers (physical and virtual controllers) in multiple virtual SD-WAN.

1.2. Research Aim and Objectives

The advent of multiple VNs to SD-WAN in 2013 (Zilong Ye et al., 2013) motivates the author to design a new algorithm to determine the number and positions of the physical and virtual controllers of multiple virtual SD-WANs. To the best of the author knowledge and from the conducted literature review, it is found that there is no study, yet which is concerned with addressing this untapped niche. Therefore, ***this thesis aims to design a novel algorithm for the controllers of the virtual networks (COVN) placement in SD-***

WAN with multiple VNs to optimise its reliability, latency, load balancing and resources utilisation.

COVN placement algorithm intends to achieve the optimal controller placement for the SD-WAN, which can be a backbone of smart cities and 5G cellular network. Simultaneously, it can obtain an acceptable controller placement for wider SD-WANs. Also, COVN placement can be applied on flow hypervisors (such as Flowvisor), if the SD-WAN uses the flow hypervisors to connect the VNs to the physical controllers. It is noteworthy that, the hypervisor is a software enables communication between the physical machines and virtual machines.

Finally, the objectives of this research are as follows:

1. To evaluate the bandwidth usage of the controller packets.
2. To measure the latency metrics of the data and control planes.
3. To evaluate the effect of the controller placement on the SD-WAN latency metrics.
4. To examine the performance of COVN placement algorithm for different: topologies; number of clusters; number of failure controllers and load of VN.
5. To evaluate the resulted latency and reliability metrics after applying COVN placement algorithm.

1.3. Research Methodology

This research has been chosen to consolidate the author knowledge about the latest network technologies such as SDN and NV, as well as to contribute to the filling in of the gap of this undergoing subject. Through the process of this research, the author focuses on learning and following the standards of these new inventions. Also, the author keeps in touch with the research communities, such as academic conferences and workshops, as well as industrial events and webinars.

The research methodology can be summarised in five processes, which are as follows:

1) Identifying the Research the Gap (Process 1)

A comprehensive investigation and literature review of the SDN and NV is conducted to identify an unsolved problem, the solution of which could be the contribution of this research. The outcome of this process is the necessity of a novel algorithm for placing the controllers in SD-WAN with multiple virtual networks.

2) Examining the Effectiveness of Controller placement on Optimising the Desired Objectives (Process 2)

This examination was carried out to answer the following questions; how and to what extent the controller placement could affect the resources utilisation and the latency optimisation. The focus of this research is to not only introduce a placement algorithm which optimises several objectives, but also prioritise these objectives according to the level of the objective's refinement which could be gained from improving the controller placement. Therefore, an intensive examination and evaluation is undertaken to discover the effectiveness of controller placements on: (A) the resources utilisation and (B) the latency optimisation, as explained below.

A) Resources utilisation: one common way to employ the controller placement for optimising the resources utilisation is by dynamically modifying the active resources according to the requirements of SD-WAN. Interestingly, the present research suggests another method which could be used to utilise the bandwidth of the network by connecting the switches to the controller through the path of the most feasible bandwidth instead of the shortest path. This has led the researcher to produce and publish a novel routing algorithm, called Shortest-Feasible OpenFlow path (SFOP). SFOP routing algorithm uses the controller statistics to compute the path of shortest-feasible bandwidth. Therefore, it does not require any extra data collection, however, it has a slightly higher computation complexity than the shortest path routing algorithm. ***The finding of this process was that the control packets do not hugely affect the network bandwidth because of its small size.*** Therefore, the researcher keeps using the shortest path between the switch and controller to maintain the smallest latency since the bandwidth is not highly degraded by switch-controller communication (see chapter 4).

B) Latency optimisation: another significant finding of this research is determining the ratio of reducing **connection control-path latency versus the connection flow-setup latency** due to optimising the controller placement. Both of these latency metrics are defined by this research (see chapter four-section 4.3.5). The ***connection flow-setup latency*** is defined as the time that is required to create a connection between the source and destination nodes and to pass the first packet between them. This latency includes both latencies of the control and the data paths. While, the ***connection control-path latency*** consists only of the latencies of the control paths for the control packets, which are required to install the flow-rules on the switches. This ratio shows to what extent the controller placement is important for optimising the connection flow-setup latency.

Surprisingly, previous researchers only compared the improved features of latency before and after optimising the placement such as switch-controller latency, inter-controller latency or the flow-setup latency (see chapter three section 3.3). However, they should have compared the results of optimisation to its final target which is the connection flow-setup latency. To perform this task, it was necessary to measure and analyse all features of the connection flow-setup latency on hardware and emulated equipment. The features of the connection flow-setup latency were extracted from two detected tests as follows:

- First test was performed to find the propagation delay of Data-Path Link (**DPL**), Data-Path Switch (**DPS**) delay and Controller Delay (**CD**) (see Figure 1.1).
- Second test was performed to compute the propagation delay of Control-Path Link (**CPL**) and Control-Path Switch (**CPS**) delay (see Figure 1.1).

It is noteworthy that , the delay is only the portion of time that is consumed at the link, switch or controller. The outcome of these two tests shows that the ***DPS*** produces a long delay (≈27 ms) due to the time needed to install and apply the forwarding rules at the first packet of a flow. Also, the controller wastes a massive delay which is about 172 ms. Whereas, the control path always spends a small delay in microseconds.

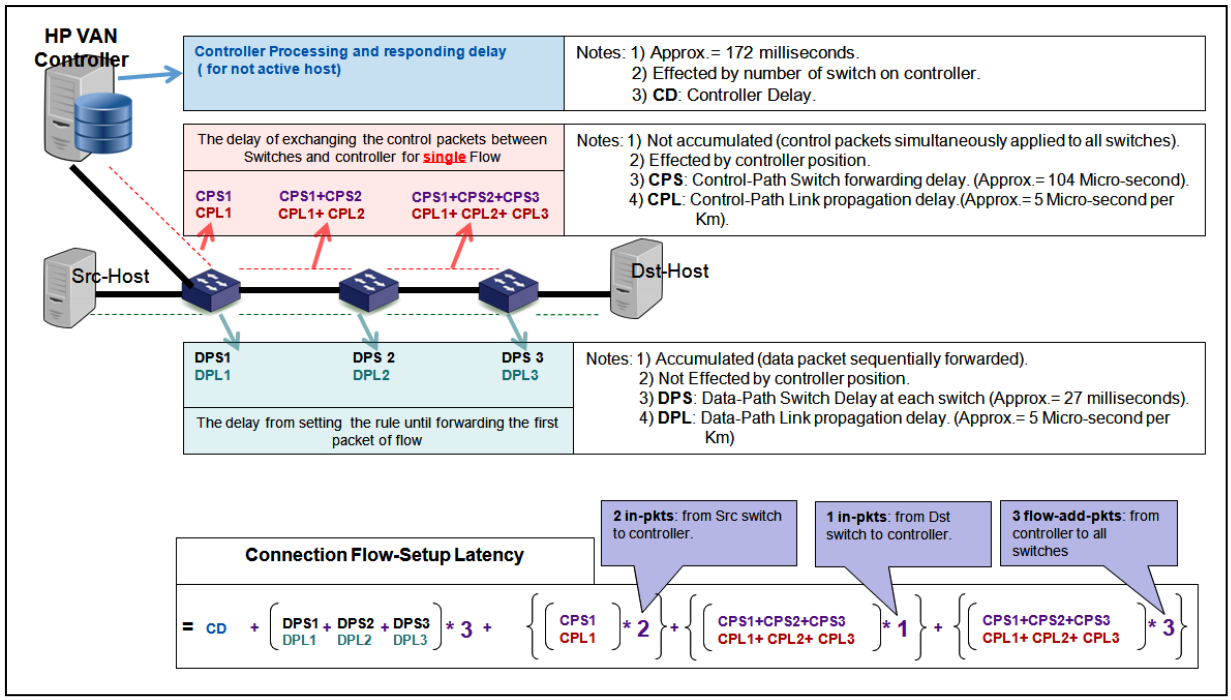


Figure 1.1: The outcome of examining the latency

Finally, these tests assist in the formulation of the equation for computing the connection flow-setup delay for the single controller as shown above in Figure 1.1 (see chapter 4). The findings of these tests demonstrate prominent issues as follows:

- The controller placement has a tiny improvement in a connection flow-setup time if it is focused only on optimising the latency of control path because it is always a small latency in comparison to the value of the connection flow-setup time.
- In the control path, the propagation latency (**CPL**) is almost smaller than the switch latency (**CPS**) for the network of a small and medium geographical expansion (campuses, cities and countries) and for networks which have most of their links being less than or equal to 20Km in length. Therefore, it is possible to find the shortest path according to the hop count rather than the distance, which reduces the computation complexity of placement algorithm tremendously, as will be explained in the design of the COVN placement.
- Even for a network with long links, it is possible to use the hop count to find the shortest path because it does not cause a noticeable deterioration to the optimisation of placement algorithm when the network links have a similar length, this will be illustrated in chapter 7.

3) Design of COVN Placement (Process 3): this process exploits the findings of process 2 to create the complete placement algorithm, as will be explained later in chapter 5.

4) Implementation Testbed (Process 4): in this stage, the COVN placement needs to be tested on SD-WAN of different sizes and scales, therefore, a new simulator was built using the Matlab to simulate the COVN algorithm. In addition, it can generate a topology, modify the virtual networks and calculate the connection flow-setup time for SD-WAN with multiple controllers in normal and failure status and find the statistic of latency and reliability when implementing the novel algorithm. After that, the Mininet emulator was adapted through this work to emulate samples of networks in order to validate the results of the simulator.

5) Placement Tests and Results (Process 5): The COVN placement was applied intensively to find the several placements on each one of the six test topologies, which were selected from real networks of the internet zoo topology database (Knight et al., 2011). These tests revealed that the COVN algorithm could achieve a good placement in terms of reliability, latency and load balance on different scale and size of topology in a very short time. After that, a comparison was performed with the results of the POCO placement algorithm to show that the COVN placement algorithm produces better placement decisions.

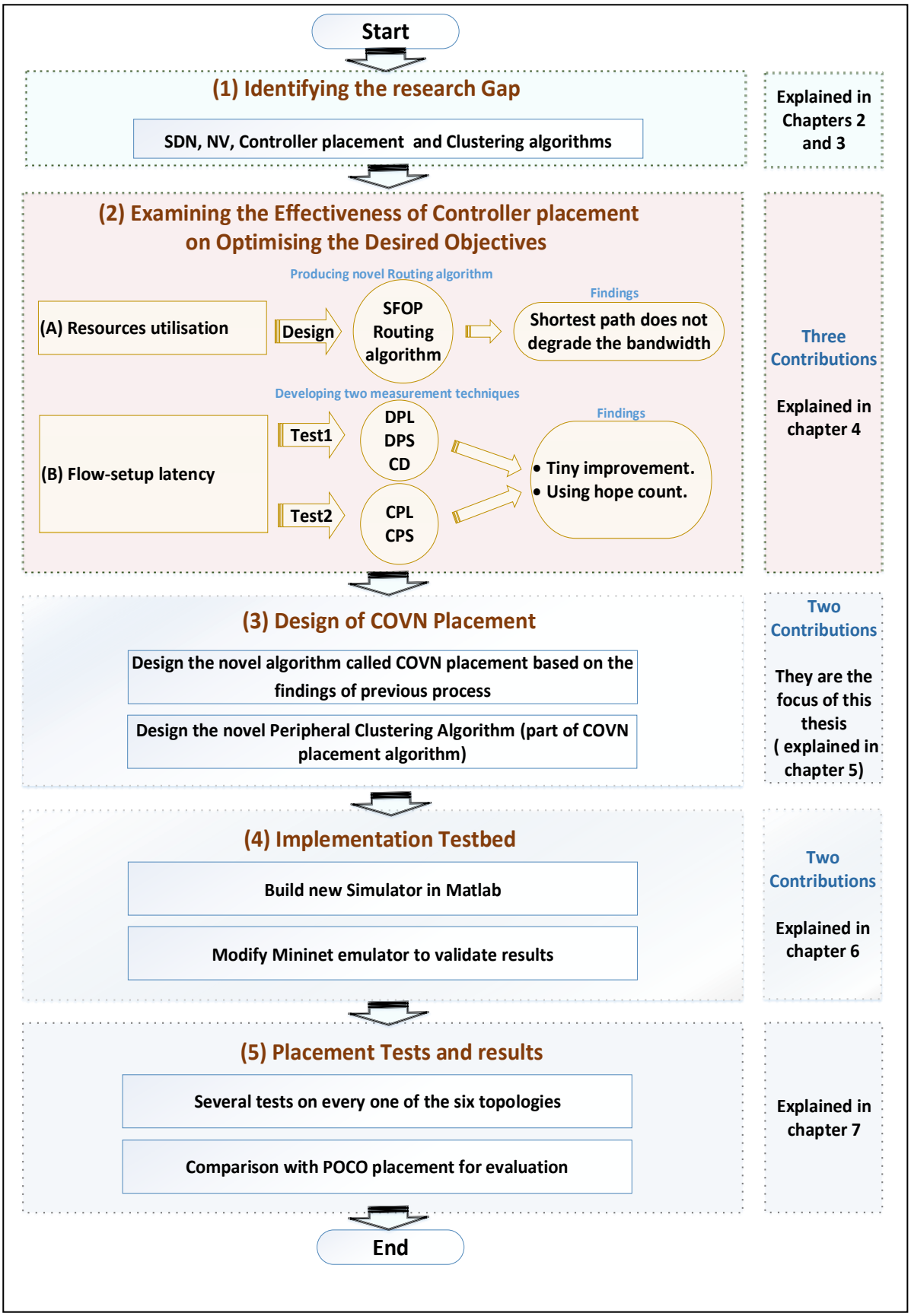


Figure 1.2: Research methodology of this thesis.

1.4. Research Contributions

This research contributes to innovate COVN placement algorithm to optimise multi-objectives simultaneously, which are: the reliability; the latency; the load balance; and the resources utilisation. The latency and the reliability are improved by the COVN placement algorithm according to the service requirements using a specific weight for every one of them.

To satisfy the requirements of this research, it was necessary to investigate, examine, and evaluate, the affected metrics before and after applying the optimisation of the controller placement on the multiple virtual SD-WANs. Accomplishing the above steps led to the following contributions (see Figure 1.2):

1. Designed and implemented a novel controller placement algorithm for multiple virtual SD-WANs, named COVN placement, to optimise their controller placement.
2. Developed a novel routing algorithm for SDN, called SFOP, to optimise the bandwidth utilisation. This published research has been impacted by several works such as "WAN Optimization to Speed up Data Transfer" (Soewito *et al.*, 2017) and optimise routing algorithm of SDN (Huddiniah *et al.*, 2018).
3. Designed and implemented the novel Peripheral Clustering Algorithm, for fast and balanced portioning of the VNs, and it is part of the COVN placement.
4. Developed two measurement techniques to measure the latencies of the data and control plan of SD-WAN.
5. Modified the Mininet emulator to implement the decision of the controller placement.
6. Built COVN simulator to:
    a) Generate the physical and virtual topology and their matrixes.
    b) Compute the COVN placement.
    c) Calculate evaluation statistic after applying the controller placement.

1.5. Practical Impact of Research Contributions

To demonstrate the impact of the research contributions, two different examples are presented in Figure 1.3. Example 1, displays the implementation of the existing controller placement algorithms on the physical SD-WAN during the day-load and night load. Example 2, shows the contribution of this research, which is the placement of physical and virtual controllers on the sliced SD-WAN. Example 2, is performed by applying the novel COVN placement algorithm on the sliced-SD-WAN in day and night loads. However, this work is the first research which provides a specific placement algorithm for the sliced SD-WAN. It is important to compare its dynamic implementation with the existing placement algorithms to clarify the differences between there outcomes.

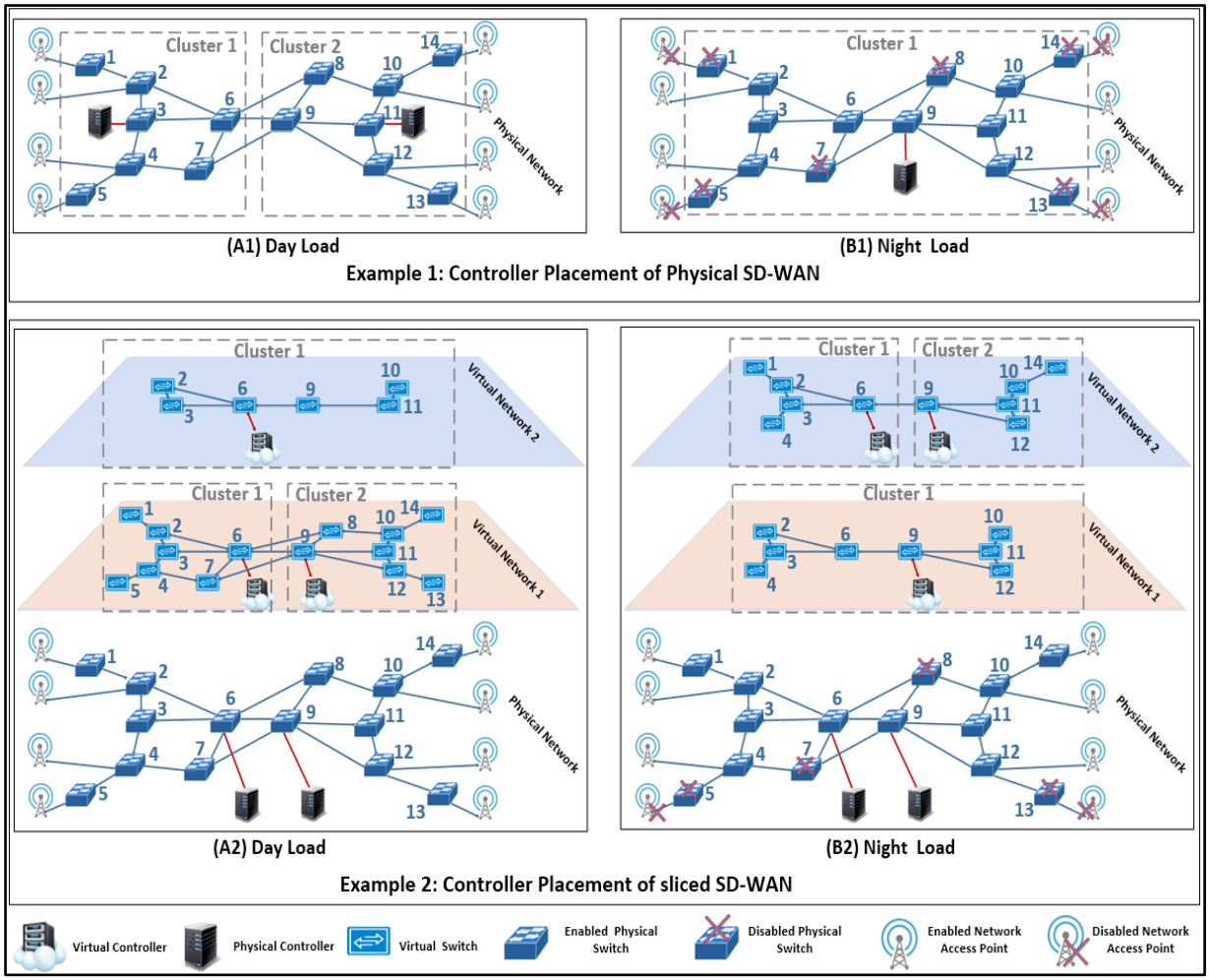


Figure 1.3: Examples of dynamic implantation for the controller placement algorithms.

The existing placement algorithms are designed only for the placement of the physical controllers, as shown in ***Example 1***. These algorithms partition the physical SD-WAN and find the locations of the physical controllers in relation to their clusters. In this

example, two clusters are produced when the network has a heavy load during the daytime (see Figure A1). While, only one cluster is created when the load of the network goes down during the night time (see Figure B1). ***From Figures A1 and B1, it is noticeable that, the existing placement algorithms need to change the locations of the physical controllers when changing the controller placement because they locate the physical controllers in the centre of every cluster. This means implementing the existing placement algorithm on the multiple VNs will lead to creating a different location of physical controllers for every VNs, which is not an applicable solution.***

Example 2, presents two VNs run over a physical SD-WAN. Every VN creates two different topologies, one at day load (see Figure A2) and second at night load (see Figure B2). VN-1 has two controllers in the day load, while it has only one controller in the night load due to the reduction of virtual switches number. VN-2 has an opposite statuses of VN-1. The second example shows that, COVN placement does not partition the physical SD-WAN when calculating the placement of the physical controllers. Instead of that, it determines the locations (nodes) which have the optimal connectivity and closeness according to the entire SD-WAN to place the physical controllers. After that, the COVN algorithm partitions every virtual network individually into clusters. Then, it finds the optimal physical controller which can host the virtual controller for every cluster. ***Separating the placement of the physical controllers from the placement of the virtual controllers in COVN placement algorithm enables it to optimise the placement of the virtual controllers for every VN dynamically without changing the locations of the physical controllers (see Figures A2 and B2).***

In addition, the COVN placement locates the physical controllers near to each other (see Figures A2 and B2), which reduces the used number of the switches and the links when exchanging control traffic and converging the control information. While, the existing placement algorithms locate the controllers in scattered locations (see Figures A1 and B1), which increase the used resources of inter-controller's communication.

Furthermore, considering the network virtualisation in this research improves the SD-WAN scalability by increasing the number of virtual resources when required. For instance, the number of the physical switches is approximately duplicated when two VNs are created in Example 2. Also, the example shows that, VN-2 increases its number of virtual switches when required at night load. At the same time, the network

virtualisation provides complete isolation for every VN which enables it to have different topologies and policies as shown in Figures A2 and B2.

From the comparison above, the practical impacts of the research contributions is creating a specified placement algorithm for the sliced SD-WAN. Also, considering the VNs in this research improves the scalability, the management flexibility and the isolation of the shared resources of the SD-WAN.

Finally, the other differences of the resulted reliability and latency metrics will be illustrated in Chapter Six.

1.6. Research Informed Teaching:

The author has had the opportunity to work alongside his supervisor Dr. Ali Al-Sherbaz to produce the SDN learning material for MSc and the final year module. This module is designed based on the theoretical and practical implementation of SDN and the outcome of this research/thesis. This SDN module is successfully running for two years.

1.7. Thesis Structure

This thesis is constructed in seven chapters. Chapter one provides the introduction to the research subject along with its aim, methodology and contributions. Chapter two briefly reports the essential concepts and background of SDN and NV, which are considered as the standard of this research. Especially as, SDN and VN are still open-ended innovations and keep up with their progress through the research journey. Also, Chapter two presents the intensive literature review of the controller placement algorithms and the clustering algorithms. This literature review clarifies the necessity for producing the new algorithms and demonstrates the disparity of the thesis contributions from the related works. Then, chapter three shows to which extent the controller placement could affect the SD-WAN bandwidth and latencies by examining and measuring them before and after applying the controller placement. This chapter illustrates the answer to the mentioned question by presenting three detected experiments. After that, the design of the COVN placement algorithm is explained in chapter four. While, chapter five demonstrates the two testbeds, which are built to implement and evaluate the COVN placement. Chapter six discusses the results of six different categories of test. Then it exposes two different methods for evaluating the COVN placement algorithm, which are: using the Mininet emulator, and comparing the results with the POCO placement algorithm. Finally, the seventh chapter concludes the findings of this research and suggests probable future work.

CHAPTER TWO
BACKGROUND AND LITERATURE REVIEW

2.1. Introduction:

This chapter presents the background of SDN, NFV and NV. It also shows the related work of controller placement algorithms and clustering algorithms. The first part defines SDN and virtualisation technology, then it demonstrates the necessity of combining them to create the multiple VNs.

The focus of this research is discussed in the second part of this chapter, which presents the literature review of the controller placement problem. The studies which dealt with solving the placement problem are investigated in Section 2.3 to give an understanding of the strengths and weaknesses of the performed works. Aiming at addressing the gap and highlighting the contribution of the present work, this part does not only showcase the apparent gap in this filed, but it also demonstrates the problems related to the existing placement algorithms. The latters are addressed through the novel placement algorithm of SD-WAN with multiple virtual networks contributes to solving them.

Another vital part of the literature review is the clustering algorithm. Section 2.4 summarises the existing clustering methods and their characteristics. This review shows the necessity for developing a novel clustering algorithm as an essential part of the novel placement algorithm.

At the end of every section, the summary and the conclusion are presented to demonstrate the purposes of each part.

2.2. Network Softwarisation

The increasing number of the deployed devices and services of the network has led to the escalation of the complexity of network deployment and management. This has resulted in the necessity for the future network to be more flexible and dynamic than the current network (Alasadi and Al-raweshidy, 2018). Meeting these requirements has given birth to the novel technologies needed to automate the network configuration, management, and maintenance. These technologies have tended to replace hardware devices with software equivalents. The network softwarisation promotes the more

flexible management, deployment, and development of network components (Lara, 2015).

2.2.1. Software Defined Network (SDN)

SDN is an emerging programmable network which has a logically centralised controller that administrates the network switches in cooperation with the management applications. This happens through a new interface between the control and data layers, namely OpenFlow protocol (Cox *et al.*, 2017).

The main reason for the emergence of SDN is that the traditional IP network has a vertical integration between the control plan (which decides how to manage the traffic) and data-plan (which forwards the traffic according to the decision of the control plan). That means both plans are bundled inside network devices (Foundation, 2015)(SDxCentral, 2015), as shown in Figure 2.1 (A). This vertical integration increases the management complexity and produces a very distributed control mechanism which lacks the global view of the network when creating the control decision.

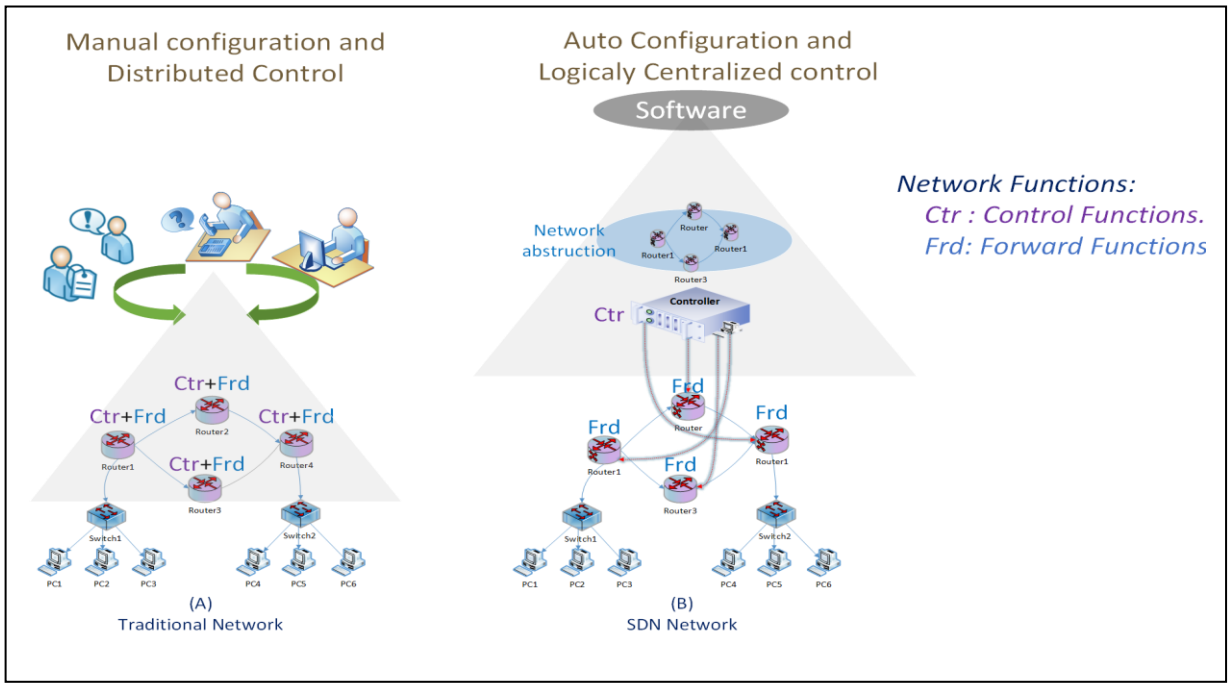


Figure 2.1: Comparison of the traditional and SDN Network

Fortunately, SDN decouples the control and data plans and migrates the controlling functionality to the central software known as a controller, (Stallings, 2013) see Figure 2.1(B). The purpose of that is breaking the vertical integration and moves the complexity away from the hardware to software. This promises for more innovation and flexibility in the software while the hardware becomes a simple device that focuses only on

forwarding traffic. Also, SDN controller provides the management applications with a real-time abstraction of network devices. That enables the management applications to configure the network devices automatically and dynamically. So, the controller simplifies the network configuration, reduces the configuration time and minimises configuration errors. In another words, the application programs can change and manage anything for the SDN to produce a programmable network.

Both the central control and the programmability of SDN allow implementing the network NV and NFV to create the overly multiple virtual networks over the physical infrastructure.

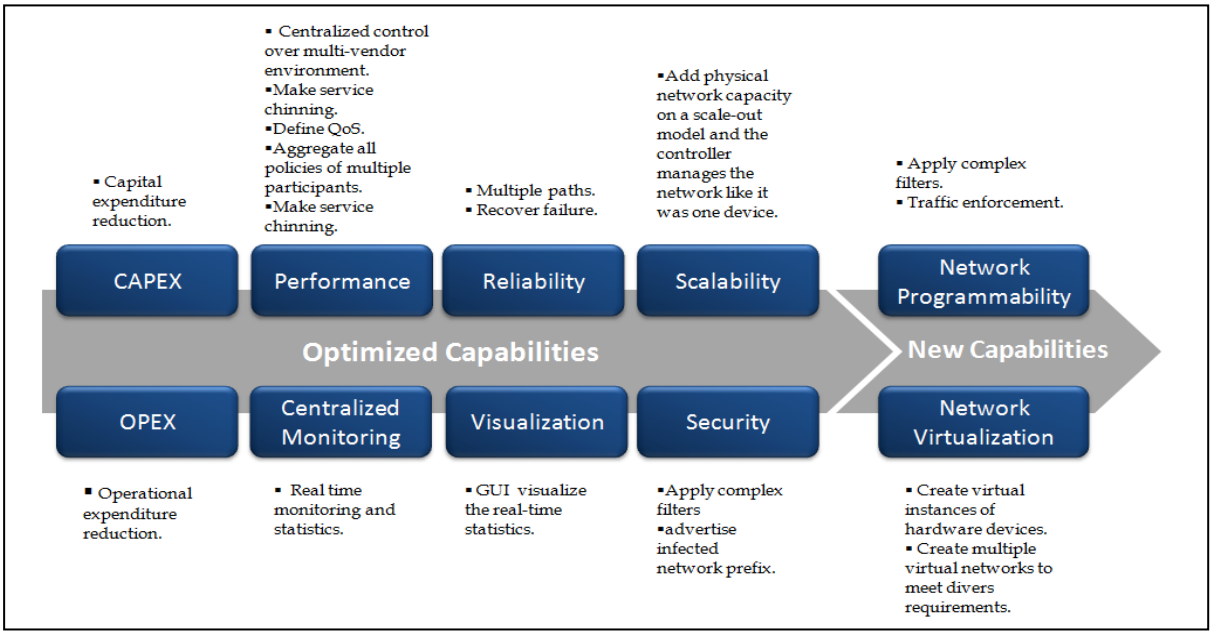


Figure 2.2: The optimisation and the new capabilities of the network after the move from traditional network to SDN.

Finally, SDN optimises some capabilities that exist in the traditional network such as the cost, performance, reliability, scalability, centralised monitoring, visualisation and security. Meanwhile, the centric-architecture of SDN creates new capabilities for the network like network programmability and network virtualisation (Cox *et al.*, 2017).

2.2.2. The Virtualization and Its Relation to SDN

The virtualisation is defined as the capability of emulating hardware components like storage, computing machines and network resources, using the software (Ibn-Khedher *et al.*, 2015). Virtualisation creates a virtualised (software) entity mimicking the exact functionality of the original hardware. Meanwhile, virtualisation decouples these entities from the underlay hardware by executing them as isolated instances of software over the Operating System (OS) of these devices. Then, it is possible to create or destroy these

entities dynamically according to the needs of the system. Also, it could duplicate or migrate these virtualised entities to recover the failure, load balancing or deploy the services dynamically (Ibn-Khedher *et al.*, 2015).

To support the flexibility and to extend the softwarisation abilities of SDN, it should be integrated with the virtualisation. Virtualisation and SDN are different concepts, but they are closely related. This correlation is a consequence of aiding the SDN to the growth and the standardisation of the Network Operating System (NOS), which is the basic ground to run the virtualisation. Furthermore, SDN feeds the centralised control, which makes the management of virtualised equipment much easier and efficient. Vice versa, virtualisation supports SDN programmability and provides powerful tools to simulate and test SDN network innovation (Nakao and Yamada, 2016).

It could categorise the virtualisation in the network into two parts, the Network Function Virtualisation (NFV) and Network Virtualisation (NV). NFV is the virtualisation technology, which allows any network device to be emulated, as a software component and is accommodated over the NOS of the hardware devices (Chiosi *et al.*, 2012). NV is the ability to abstract the connectivity of the logical network (network services or virtual components) to create the overlay network and decoupling it from the underlay hardware network (Rao, 2014). Therefore, NFV creates the virtual components, while the NV responsible for connecting these components to construct the logical slice of the network (Cox *et al.*, 2017). In another words, both are required to construct a fully virtualised (logical) SDN network, see Figure 2.3.

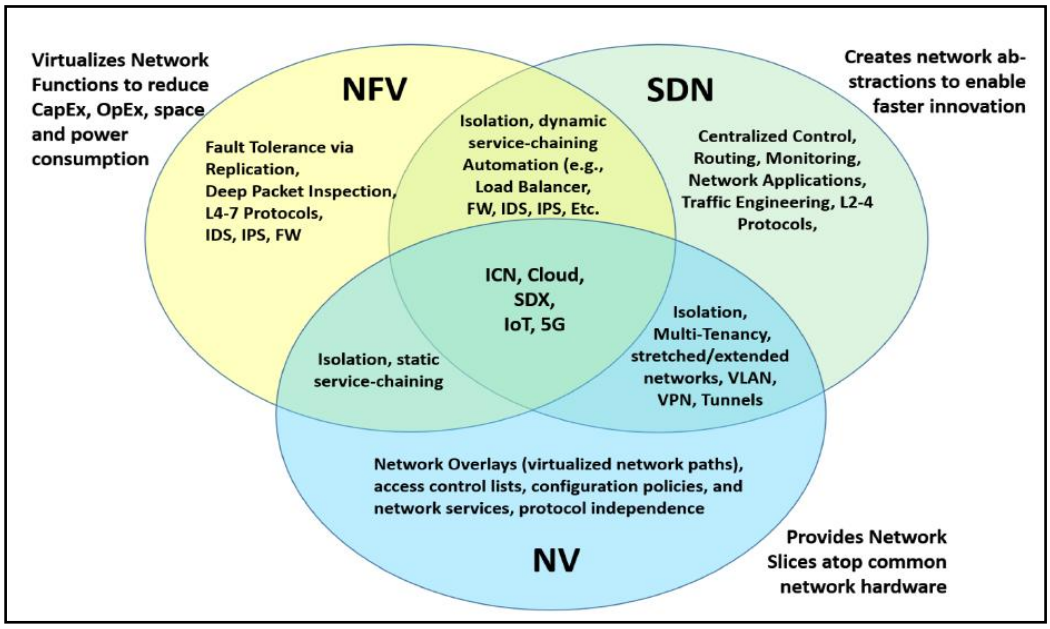


Figure 2.3: The relationship between SDN, NV and NFV(Cox *et al.*, 2017).

They enable SDN to: 1) Share a resource; 2) Isolate the shared resource; 3) Aggregate multiple virtualised resources; 4) Dynamically create and destroy the resources; and 5) Extend the management capabilities like migrating a resource (Jain and Paul, 2013). The combination of SDN, NFV and NV are exploited to serve many implementations of the future internet such Information-Centric Networking (ICN), Internet of things (IoT) and the backbone of the 5G mobile network (Service Providers, 2017).

For example, NV and NFV provide multi-tenancy in the data centre (Fekih Ahmed *et al.*, 2015). Furthermore, SDN enables NV to connect (chaining) network services (NFV) according to the requirement of network applications (Mijumbi *et al.*, 2016). Finally, attaching the SDN and NFV to NV improves its limitation for recovering the failure, disaster and big events (Rao, 2014), see Figure 2.4.

Figure 2.4 presents three VN's which serve three different sectors. Part (A) displays the VN under normal traffic, while part (B) illustrates how the network could respond to recover the troubles and keep a reliable performance by adding the required virtual nodes and links. These capabilities lead vendors and network operators to apply this integration to SD-WAN.

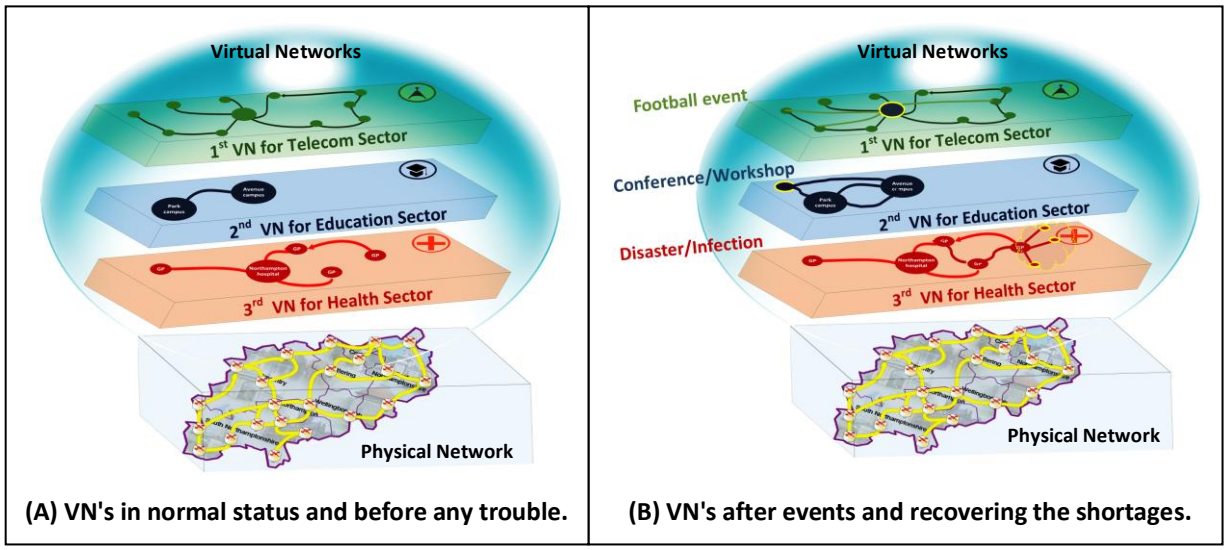


Figure 2.4: The integration of SDN, NFV and NV, which improves the network capability for recovering the failure, disaster and big events by enhancing the virtual resources.

2.3. SDN Controller Placement Algorithms

Optimizing the controller placement of SD-WAN for multiple virtual networks is the main aim of this research. Therefore, a comprehensive literature review of the controller placement problem is performed in order to: 1) Identify the gap and show the studies

which have been conducted in relation to this problem; 2) Use the findings of the previous work as a starting point after the practical and analytical evaluation; 3) Prove the experimental findings of the present research by highlighting the findings of other studies.

This research has classified the controller placement algorithms according to the optimised criteria. The optimised criteria could be arranged into five groups as follows: a) Minimizing network latency; b) Maximizing resilience and reliability; c) Load balancing; d) Decreasing infrastructure cost and energy consumption; and e) Multi-objectives approach, as demonstrated in Figure 2.5.

The following subsections present the related works of the placement algorithms regarding their optimised criteria. It also highlights the findings which serve the direction of this thesis. The highlighted outcomes are summarised as a Table 2.6 at the end of Section 2.3.6. summary and conclusion.

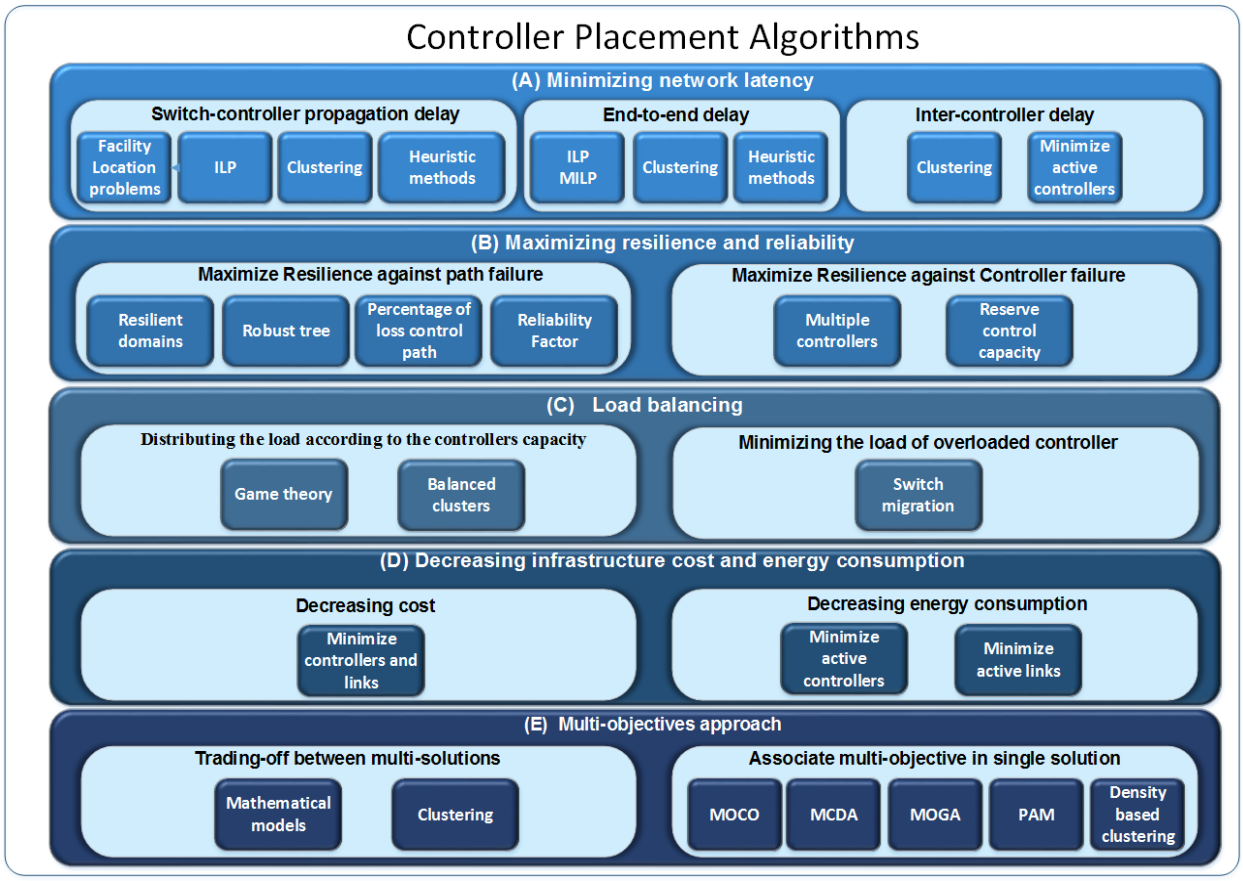


Figure 2.5: The proposed classification of the controller placement algorithms according to the optimised criteria.

The figure presents each class as three layers of rectangles: (1) The main rectangle represents the optimised criteria, (2) Sub-rectangles display the branches of optimised criteria and (3) The Internal rectangles show the optimisation methods used in each sub-class.

2.3.1. Minimising Network Latency

The controller placement in the network does not influence the latency of all packets of the flow; it impacts only the latency of its first packet, which is called the end-to-end flow-setup time. The end-to-end flow-setup time is defined as the time consumed since sending the first packet from the source until it is received at the destination. It involves all the time required to request and install the rules into all the switches on the path from the source to the destination, added to it the latencies for forwarding the packet on that path such as propagation, transmission and queuing latencies (Zeng *et al.*, 2015) (Guodong Wang et al., 2017) (He *et al.*, 2017).

The researchers that optimise the controller placement, based on the latency as a primary objective can be sorted as three groups, namely: 1) switch-controller propagation delay; 2) end-to-end delay; and 3) inter-controller delay, see Table 2.1. In the following, the evolution of these groups over time will be traced.

Class No.	Sub-optimization criteria	Publication No.	Reference	Method
1	Switch-controller propagation delay	1	(Heller et al., 2012)	Facility location problem
		2	(Bari *et al.*, 2013)	ILP
		3	(Penna *et al.*, 2014)	Modifies k-clustering
		4 5	(Tuncer et al., 2015a) (Tuncer et al., 2015b)	Clustering $P_{cluster}$ and P_{weight}
		6	(Guodong Wang *et al.*, 2016)	Improved K-Means
		7	(Zhao et al., 2017)	Exemplar-clustering
		8	(Sahoo et al., 2017)	Particle Swarm Optimization and Firefly algorithms
2	End-to-end delay	9	(Zeng et al., 2015)	ILP
		10	(Guodong Wang *et al.*, 2017)	Optimized K-Mean algorithm
		11	(He et al., 2017)	Spectral clustering and MIP
		12	(Sood and Xiang, 2017)	Analytical model
3	Inter-controller delay	13	(Nagano and Shinomiya, 2015)	Clustering
		14	(Zhang et al., 2016)	Analytical model.
		15	(Han et al., 2016)	Exhaustive search algorithm
		16	(Zhu et al., 2017)	Adapted K-Mean
		17	(Li *et al.*, 2018)	Approximation algorithm

Table 2.1: The sub-classes of the publications that place the controllers to minimise the SD-WAN latency.

The first group of researchers assume that the optimal controller placement is the one that reduces the propagation delay between the switches and the controllers. Investigating the effect of switch-controller delay on the flow-setup time starts by Heller

et al. (2012). Heller uses the exhaustive search for all possibilities to find the optimal solution, which produces high computational complexity. ***The critical finding stated by Heller is that one controller provides sufficient reaction-time but cannot satisfy fault tolerance requirements.*** Heller is followed by similar works such as (Bari *et al.,* 2013) and (Penna *et al.,* 2014). Two other works, propose to minimise the hop count from the controller to the managed switches (Tuncer *et al.*, 2015a) (Tuncer *et al.*, 2015b). Also, some research used the clustering to keep minimal propagation latency (Guodong Wang *et al.,* 2016).

Finally, the authors in (Sahoo *et al.*, 2017) develops a heuristic frameworks to optimise the switch-controller latency in smaller computational complexity. The first group's research missed the other components of latency in the process of installing the flow rules from the controllers to the switches.

The second group aims to minimise the end-to-end delay between the switch and the controller to find the optimal controller placement. The end-to-end delay is composed of many latencies like, link propagation, switch transmission, controller processing and queuing latency (Wang *et al.*, 2018). The studies in (Zeng *et al.*, 2015) (Wang *et al.*, 2018) and (He *et al.*, 2017) optimise a different composition of latencies, which formulate the end-to-end delay. Subsequently, another piece of research (Sood and Xiang, 2017), suggests changing the controller placement problem into controller selection problem. Also, ***it infer to an important fact, which states, that it is vital to adapt the logical controllers dynamically using a topology-independent and a low complexity placement algorithm.*** The second group presents a good placement for multi dedicated domains but demonstrates a poor placement for a logically centralised control layer due to missing the inter-controller latency.

The third group creates a wider view of end-to-end delay. They take into account the inter-controller delay in addition to the switch-controller delays. For example, Nagano and Shinomiya (2015) prove that the controllers' inter-domain communication negatively affects the network performance. Also, Zhang *et al.*, (2016) states two important facts which are: ***1) The reaction time of controller is affected by the consistency algorithm applied between the controllers such as Raft or anti-entropy consensus and the level of consistency; 2) The inter-controller delay should be added to the end-to-end delays.*** This is followed by three algorithms mitigate the controller-

controller delay and the switches-controller delay in (Han *et al.*, 2016) (Zhu *et al.*, 2017) and (Li *et al.*, 2018). The studies in the third group provide more realistic results, which demonstrate a better controller placement according to the latency.

2.3.2. Maximising Resilience and Reliability

Failing in the installation of the flow rules into the switches results in halting the network traffic, which is worse than delaying the installation of these rules (Liu *et al.*, 2016). Therefore, the controller placement problem is established for maximising the resilience against any failure in the control layer as well as increasing its reliability (Zhang *et al.*, 2011).

Class No.	Sub-optimization criteria	Publication No.	Reference	Method
1	Maximize Resilience against path failure	1	(Zhang *et al.*, 2011)	Minicut clustering
		2 3	(Xiao et al., 2014) (Xiao *et al.*, 2016)	Spectral clustering K-self-adaptive clustering
		4	(Aoki and Shinomiya, 2016)	Clustering
		5 6	(Jimenez et al., 2013) (Jimenez et al., 2014)	Robust tree (K-Critical algorithm)
		7	(Guo and Bhattacharya, 2013)	Robust tree (clustering)
		8 9	(Hu et al., 2013) (Hu et al., 2014)	Percentage of loss control path (Simulated Annealing)
		10	(Liu et al., 2016)	Reliability Factor (K-Mean)
2	Maximize Resilience against Controller failure	11	(Muller *et al.*, 2014)	Redundant controllers (ILP)
		12	(Perrot and Reynaud, 2016)	Redundant controllers (ILP)
		13	(Tanha et al., 2016)	Redundant controllers (Exhaustive search)
		14 15	(Killi and Rao, 2016) (Killi and Rao, 2017)	Redundant controllers (MILP)(Simulated Annealing)
		16	(Abdelaziz et al., 2017)	Capacitated clustering

Table 2.2: The sub-classes of the publications that place the controllers to maximising resilience and reliability of SD-WAN.

Different studies have been done on the optimisation of the controller placement according to the resilience and the reliability of SD-WAN, these studies can be categorised into two types. The first type focuses on the maximisation of the resilience against connection failure between switches and controllers, whereas the emphasis of the second type is on increasing the reliability versus control failures including the controller failure, see Table 2.2.

In the former, many researches explore the possibility of using the clustering methods to produce homogenous domains. Initially, Zhang *et al.* (2011) explore the effect of controller placement on the resilience of link failure between the switches and the

controllers. Following that, the concept of clustering the network into resilient domains is optimised in (Xiao *et al.*, 2014) and (Xiao *et al.*, 2016). ***An interesting outcome of last two researches is that, expanding the SD-WAN pointedly amplifies the inter-controller broadcast storm, however they did not intend to solve this problem.*** On this line of thought, the works in (Nagano and Shinomiya, 2015) and (Aoki and Shinomiya, 2016) optimises the network resilience using the partitioning. ***The disadvantage of resilient clustering placements is not ideally balanced with all clusters.*** There is another method to tolerate the path failure that suggests a model of robust control tree, which is presented in (Jimenez *et al.*, 2013), (Jimenez *et al.*, 2014) and (Guo and Bhattacharya, 2013). The last three methods may not probably work for nested networks owing to the high computational complexity of searching the optimal tree. The other researchers of path failure in (Hu *et al.*, 2013) and (Hu *et al.*, 2014) propose a new metric, which is the percentage of loss control path, to maximise the reliability of the control layer. ***Hu et al. (2014) infer that the controller placement could improve the reliability without creating unsatisfactory latencies between switches and controllers.*** The last method identified to overcome links outage is proposed in (Liu *et al.*, 2016). Liu uses the average distance of multi-paths between the switches and controllers as a Reliability Factor (RF). All of the mentioned studies consider only the links and nodes failure and miss the controller failures.

The second type of reliable controller placement dealt with recovering most failures of the control layer including the controller failure. The algorithms in (Muller *et al.,* 2014), (Perrot and Reynaud, 2016) (Tanha *et al.*, 2016) (Killi and Rao, 2016) and (Killi and Rao, 2017) employ a single backup controller. While, The authors in (Abdelaziz *et al.*, 2017), exploit a group of three controllers, which are assigned for every cluster to recover the controller-failure. The second type of resilient controller placement is more immune to network failures.

2.3.3. Load Balancing

Stabilising the controller's load is the key to maintaining acceptable network performance. Therefore, researchers are interested in the network of unstable load aim to dynamically balance the controller's load. Balancing the load could be achieved by distributing the load according to the controllers' capacity or minimising it in the overloaded controller, by migrating the switch to another controller, see Table 2.3.

Class No.	Sub-optimization criteria	Publication No.	Reference	Method
1	Distribute load equally	1	(Rath *et al.*, 2014)	Non-zero game theory
		2	(Yao et al., 2014)	Capacitated K-center
		3	(Aoki et al., 2015)	Minimum cut clustering Conductance clustering Distance-k cliques clustering
		4	(Aoki and Shinomiya,2015)	Conductance clustering, Spectral clustering, Betweenness centrality clustering Repeated bisection clustering
		5	(Sanner et al., 2016)	Adapted the K-Mean Hierarchical clustering
2	Minimize the load of the overloaded controller (Switch migration)	6	(Yao *et al.*, 2015)	k-way clustering and dynamic balancing algorithms
		7	(Hegde et al., 2017)	Exhaustive search and dynamic balancing algorithms

Table 2.3: The sub-classes of the publications that create balanced controller placement by distributing the load on controllers according to their capacity.

Rath *et al.* (2014) exploit the non-zero game theory algorithm to dynamically change the controller placement. The researchers in (Yao *et al.*, 2014) (Aoki *et al.*, 2015) (Aoki and Shinomiya, 2015) and (Sanner *et al.*, 2016), study graph partitioning methods to create balanced clusters. ***The clustering algorithms which were investigated in the previous publications may accomplish the load balancing of the controller with the static status of the network (load and topology). Although, it is difficult to achieve load balancing for the dynamic status of network due to the high computational complexity of these algorithms.***

Another pair of related works in (Yao *et al.,* 2015) and (Hegde *et al.,* 2017), prefer to set the controllers in fixed locations while migrating the switches among them to load their balance. ***The idea of fixing the controllers locations and offloading the switch from the overloaded controller to the relaxed one, is efficient in the case of limited changes in network load. However, it is not sufficient to recover the big changes in network load or network topology. Specifically, in the last two works where the locations of controllers are assigned according to the shape of clusters which could vary with different statuses of the network.***

2.3.4. Decreasing Infrastructure Cost and Energy Consumption

To help in deploying and operating the SDN, its CAPEX and OPEX should be reduced as much as possible to make it applicable from the economical point of view. Maximizing

the utilisation of the network resources with avoiding the degrading of its performance led to reducing the constricting and operating cost as well as the energy consumption. Therefore, several researchers inspect the possibility of inspiring a cost aware controller placement, which maximises the controller utilisation and reduces the active controllers, see Table 2.4.

Class No.	Sub-optimization criteria	Publication No.	Reference	Method
1	Decreasing cost	1 2	(Sallahi and St-Hilaire, 2015) (Sallahi and St-Hilaire, 2017)	Exhaustive search algorithms.
2	Decreasing energy consumption	3 4	(Auroux et al., 2014) (Auroux et al., 2015)	Minimize active controllers (Select optimal placement among limited locations)
		5	(Ruiz-Rivera et al., 2015)	Minimize active Links (BiP)
		6	(Hu et al., 2017)	Minimize active Links (genetic heuristic algorithm)

Table 2.4: The sub-classes of the publications that create controller placement which reduces the cost and power consumption of SD-WAN.

The authors in (Sallahi and St-Hilaire, 2015) and (Sallahi and St-Hilaire, 2017), formulate a mathematical model to find the cost-effective controller placement for a newly constructed or the updated network. It efficiently improves network cost; however, it does not take into account the effect of inter-controller delay when placing the controllers and suffers from a very high computational complexity.

Further, several placement algorithms manage the minimisation of the consumed power by minifying the active controllers (Auroux et al., 2014) (Auroux et al., 2015) (Ruiz-Rivera et al., 2015) (Hu et al., 2017). They are also causing a higher latency, in addition, they ignore the inter-controller latency.

2.3.5. Multi-Objective Approach

The last category of scholars believes that one objective (latency, reliability, load, energy or cost) cannot achieve optimal placement. Therefore, they regard multi-objectives in two ways: the first one is by trading-off between objectives; and the second is by combining multi-objectives in single placement with adaptable weights, see Table 2.5.

The trading-off between objectives is started by Hock et al. (2013). Hock investigates controller placements regarding different metrics namely, the latency between switch and controller, Inter-controller latency, resilience against path failure, resilience against

controller failure and load balancing. He infers that there is no single optimal solution, but trading-off between them is required to satisfy different purposes.

Class No.	Sub-optimization criteria	Publication No.	Reference	Method
1	Trading-off between multi-solutions	1 2 3	(Hock *et al.*, 2013) (Hock, Gebert, et al., 2014) (Hock, Hartmann, et al., 2014)	k-centres and k-median algorithms
		4	(Lange, Gebert, Zinner, *et al.*, 2015)	Heuristic approach
		5	(Lange, Gebert, Spoerhase, et al., 2015)	Pareto capacitated k-Medoids
		6	(Naning et al., 2016)	Analytical model
		7	(Hollinghurst et al., 2016)	Adapted k-means++ Local search Linear programming Full search algorithms
2	Associate multi-objective in single solution	8	(Jalili et al., 2015)	MOCO
		9	(Borcoci et al., 2015)	MCDA
		10	(Hu Bo et al., 2016)	MOGA
		11	(Liao et al., 2017)	Density-based clustering
		12	(Bannour et al., 2017)	(NSGA-II) and (PAM)
		13	(Zhang *et al.*, 2018)	MOCP
		14	(Kuang *et al.*, 2018)	Hierarchical K-Means
		15	(Tanha *et al.*, 2018)	Heuristic approach

Table 2.5: The sub-classes of the publications that consider multi-objectives to find the optimal controller placement of SD-WAN.

Hock solves all types of placement using the Brute-force search, which has NP-hard complexity. Furthermore, Hock introduces Pareto-based Optimal COntroller placement (POCO) tool. POCO Matlab framework is extended to have a Graphical User Interface (GUI) to simplify presenting the results of placement in the dynamic condition in (Hock, Gebert, et al., 2014). Subsequently, (Hock, Hartmann, et al., 2014) in their study, combine POCO tool with PLanteLab to improve the adapted placement of controller in a dynamic environment. Furthermore, a project in (Lange, Gebert, Zinner, et al., 2015) provides a heuristic approach for the placement of POCO tool. POCO algorithm could be considered as a feasible solution for small and medium networks (less than fifty switches and seven controllers). While, it is not applicable for large networks or dynamic implementations due to the high resources and time usage. The proposed heuristic method reduces the complexity of POCO placement with less accurate but acceptable results (Lange, Gebert, Zinner, et al., 2015). Consequently, when using the heuristic algorithm, a larger number

of controllers could be placed in a larger size of networks (about fifty switches and fifteen controllers). In a further study by Lange, Gebert, Spoerhase, et al., (2015), the previous heuristic algorithm is evolved to gain more accurate placement. However, the size of networks and the number of controllers in the conducted test is similar to the tests of the POCO tools, which does not really present a large network. Also, the work in (Naning et al., 2016) presents an analytical study using the POCO algorithm. It concludes that placing the controllers according to resilience satisfies the system requirements like reliability, latency and load balance. The last research in trading-off approach (Hollinghurst et al., 2016) shows that the local search and k-means++ methods are more scalable for large network. In general, the disadvantages in the above works are as follows: trading-off between many placements is not the applicable solution, the controller overload is not considered when reassigning the switches to the closest controller in failure case, and finally, the latency between the switch and controller is represented only by the propagation latency.

Moving on to the second approach which associates multi-objectives. For example the studies in (Jalili *et al.*, 2015) (Borcoci *et al.*, 2015) and (Hu Bo *et al.*, 2016) apply the multi-objective placement using the weight (threshold) of objectives to adjust the administration preferences. A different method is presented in (Liao *et al.*, 2017). Liao innovates a new density based clustering method to place the controllers according multi-objectives such as the latency, balance and reliability. ***However, it considers only propagation latency and does not really reduce the inter-controller latency. Also, implementing a capacity aware clustering produces some clusters with disjoint nodes.*** One of the latest conference papers (Bannour *et al.*, 2017), ***illustrates that modelling the traffic among controllers is essential for a consistent controller placement.*** Finally, several studies emphasize on the importance of optimising the reliability, the load-balance and the low switch-controller latency (Zhang *et al.*, 2018)(Kuang *et al.*, 2018)(Tanha *et al.*, 2018).

2.3.6. Summary and Conclusion of Controller Placement Algorithm

The literature review disclosed the gaps and their proposed solutions in controller placement problem up to the present. Consequently, it contributes to declare an unobserved niche, which is detected and solved in the present thesis. Another additional purpose is highlighting the findings of some interesting research to be evaluated and

considered in this work and support the outcome of the thesis. The presented research shows that, the controller placement has been developed to optimise the latency, reliability, load balance, cost and energy consumption of SD-WAN. The optimisation of flow-setup latency starts by minimising the propagation delay, then the end-to-end delay and ends by investigating the effect of inter-controller delay. However, minimising the inter-controller delay is poorly investigated in several research studies. Similarly, the reliability receives a good interest because it is an essential factor to maintain the work of the network. Maximizing the reliability is developed against the path and controller failure, though, the path failure has more interest than the controllers one. In addition, many studies intend to place the controller to optimise the load balancing to stabilise the network performance and enhance QoS. A few Other works consider the network cost and power consumption, which are vital to economic aspects. Finally, much research optimised the controller placement problem in the light of multi-objectives, either trading-off among them or associating them together in a different ratio of impact (different weight).

It is clear from the above summary that the controller placement problem is fairly investigated for SD-WAN. However, there is no single placement which could provide an ultimate solution for all implementations. ***In addition, most placement studies either have an algorithm with NP-hard computational complexity or use a heuristic approach with approximated results.*** Also, it is noticeable that no single placement research gave attention to an inevitable deployment of SD-WAN with multiple virtual networks. ***This means there is no placement model designed to place the controllers in this type of network. Therefore, the present thesis identifies this gap and aims to design a novel placement algorithm to optimise the controller placement for SD-WAN with multiple virtual networks, in term of maximising the network reliability and minimising its latency, as the main goal of this research.***

Finally, it is worth mentioning that while conducting the present research around 2014, some secondary gaps were not yet considered by controller placement researchers such as the effect of inter-controller delay and switch transmission delay. Therefore, this thesis investigates these gaps and includes them in the proposed novel placement algorithm. This thesis uses the findings from other studies, which have been undertaken during the time of this research, to confirm and consolidate the thesis findings, as summarised below in Table 2.6.

No.	Finding	References	Section
1	The majority of SD-WAN has a logically centralised control layer.	(Kreutz et al., 2015) (ONF, 2016) (Perrot and Reynaud, 2016) (He et al., 2017)	**2.3.1.**
2	The end-to-end delay is composed of many latencies like, link propagation, switch transmission, controller processing and queuing latency.	(Zeng et al., 2015) (Guodong Wang *et al.*, 2017) (He et al., 2017)	**2.3.1.**
3	The logically centralised control layer should consider the inter-controller communication delay (contribution in install flow rules along the path).	(Zhang et al., 2016) (Zhu et al., 2017)	**2.3.1.**
		(Liao *et al.*, 2017) (Bannour *et al.*, 2017)	**2.3.5.**
4	The controllers' inter-domain communication negatively affects the network performance (achieve consistency).	(Nagano and Shinomiya, 2015) (Zhang et al., 2016) (Han et al., 2016)	**2.3.1.**
		(Bannour et al., 2017)	**2.3.5.**
5	The exhaustive search for all possibilities produces high computational complexity.	(Heller et al., 2012) (Bari et al. 2013) (Sahoo et al., 2017)	**2.3.1.**
6	One controller provides sufficient reaction-time, but cannot satisfy fault tolerance requirements	(Heller et al., 2012)	**2.3.1.**
7	Some research represents the path latency as hope count (without justifying the reasons).	(Tuncer et al., 2015a) (Tuncer et al., 2015b)	**2.3.1.**
8	The closer controllers provide a lower end-to-end delay	(Zhang et al., 2016)	**2.3.1.**
9	Expanding the SD-WAN pointedly amplifies the inter-controller broadcast storm.	(Xiao et al., 2014)	**2.3.2.**
10	Resilient controller placement could improve the reliability without creating unsatisfactory latencies between switches and controllers.	(Hu et al. ,2014)	**2.3.2.**
11	The presented clustering algorithm may accomplish the load balancing of the controller with the static status of the network, but it is difficult to achieve it for the dynamic status of network due to the high computational complexity.	This thesis	**2.3.2.**
12	The idea of fixing the controllers locations and offload the switch from the overloaded controller to the relaxed one, is efficient in the case of a limited change in network load. However, it is not sufficient to recover the big changes in network load or network topology. Especially, in the placement which assigns the locations of controllers according to the shape of clusters, which could vary with different status of the network.	This thesis	**2.3.2.**
13	The proposed heuristic method reduces the complexity of POCO placement with less accurate but acceptable results.	(Lange, Gebert, Zinner, *et al.*, 2015) (Lange, Gebert, Spoerhase, et al., 2015) (Hu et al., 2017)	**2.3.5.**

Table 2.6: Some interesting findings from other works that support the outcomes of the thesis.

2.4. Clustering Algorithms

The proposed placement algorithm to optimise the controller placement for SD-WAN with multiple virtual networks has a novel clustering algorithm as a second part of the algorithm procedure. The new clustering aims to improve the regional distribution of the switches, load balancing among all clusters and low computational complexity. These aims are not achieved by the existing clustering algorithms. To demonstrate the necessity of the novel clustering method, a comprehensive literature review is performed to show the existing clustering algorithms and to illustrate their incompatibility with the objectives of the required clustering in this research.

Clustering is an unsupervised learning method which groups the objects into reasonable clusters. The objects within a cluster are more similar to each other than the objects in other clusters, according to a certain measure (Tseng and Yang, 2001).

This research classified the clustering algorithms into five methods (See Figure 2.6): partitioning, hierarchical, density-based, grid-based (Lu, 2003) and graph-based (Xu and WunschII, 2005) methods. The literature review of clustering lists the commonly used methods and shows their mismatching with the requirements of the proposed placement algorithm.

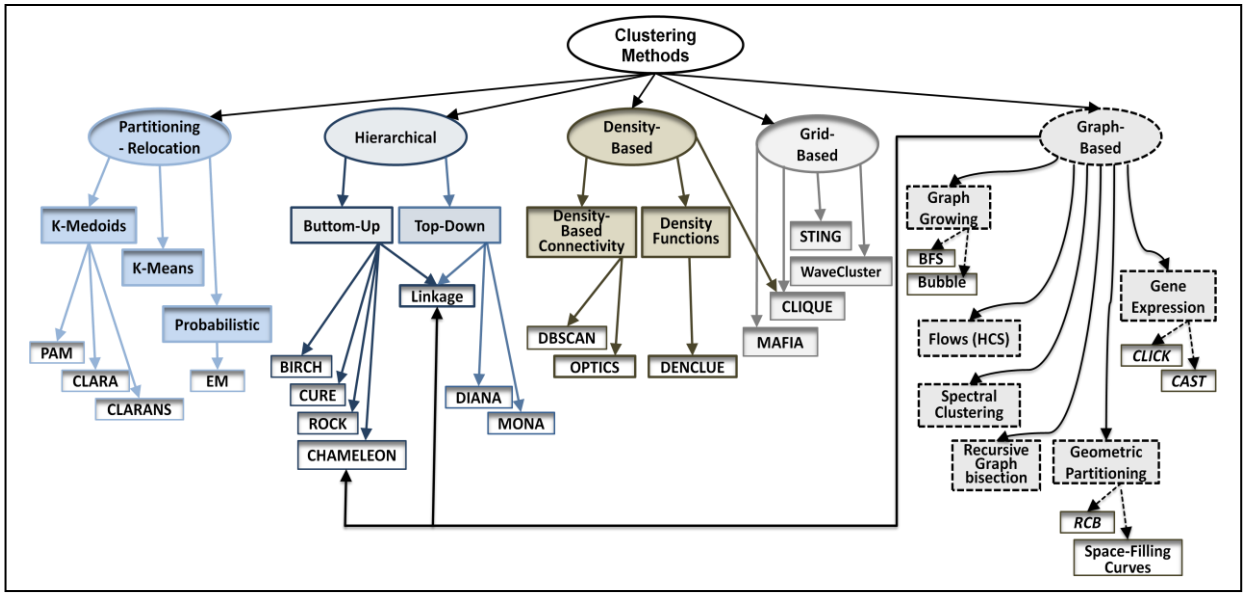


Figure 2.6: The hierarchical classification of commonly used clustering methods.

2.4.1. Partitioning Relocated Methods

The partitioning relocated methods, groups the objects into subsets (clusters) based on gravity (distance) from certain centres (centroids). First, the algorithm randomly initiates the clusters. Second, the algorithm uses an ***iterative optimisation*** to improve

the clusters by ***revising all objects*** in order to ***relocate*** them to the new centres (Berkhin, 2006) (Saraee *et al.*, 2007). It is possible to categorise the partitioning relocated methods into ***k-medoids***, ***k-means*** and ***probabilistic*** methods (Berkhin, 2006) (Saraee *et al.*, 2007).

A) k-Medoids methods: The earliest two version of K-Mediods methods is PAM (Partitioning Around Medoids) and CLARA (Clustering LARge Applications) (Kaufman and Rousseeuw, 2009). Followed by, CLARANS (Clustering Large Applications based upon RANdomized Search) Which is directed towards clustering in spatial databases (Ng and Han, 1994). ***The k-Medoids methods are low-sensitive to outliers objects and convent for all attribute types*** (Berkhin, 2006) (Saraee *et al.*, 2007).

B) k-Mean method: The K-mean has adapted in many ways to cluster the SD-WAN regarding different objectives, but it always suffers from a high computational complexity. It is also highly-sensitive and negatively affected by an outlier (Saraee et al., 2007). Another considered con for K-Means is that the method lacks stability and produces unbalanced clusters (Lu, 2003)(Berkhin, 2006).

C) Probabilistic method: Log-likelihood used as an objective function in the Expectation-Maximization (EM) method (Dempster *et al.*, 1977), which is the most popular version of the probabilistic method. ***This clustering method lacks high computational complexity and is similar to K-mean in creating unbalanced clusters***.

2.4.2. Hierarchical Methods

Hierarchical clustering methods construct a cluster hierarchy as a tree of clusters, which is called a **dendrogram**. Hierarchical methods could be categorised in general into agglomerative (bottom-up) and divisive (top-down) (Dempster *et al.*, 1977).

A) Agglomerative (bottom-up): Many techniques have been developed in agglomerative clustering. It starts by the Linkage Metrics (Xu and WunschII, 2005). Then, BIRCH (Zhang et al., 1996) is developed to minimise computational complexity and reduce the effect of the noise (outlier). Finally, CURE (Guha et al., 1998), ROCK (Guha et al., 2000) and CHAMELEON (Karypis et al., 1999) are evolved to create the arbitrary shape of clusters. ***The performance of static clustering methods can be degraded if the selected parameters are incorrect***. ***In addition, the agglomerative clustering could not be enforced to generate a specific number of balanced clusters.***

B) <u>Divisive (top-down)</u>: One of the well-known divisive methods is DIANA (DIvisive ANAlysis Clustering) (Everitt *et al.*, 2011). Another Divisive clustering is MONA (MONothetic Analysis) method (Kaufman and Rousseeuw, 2009). ***Both are not supporting the multi-objective clustering, in addition, they are not popular in practice due to thier expensive computational complexity***.

2.4.3. Density-Based Methods

The Density-Based methods construct the cluster according to the ***spatial density***, where the cluster always extends towards the direction of densely distributed objects. Consequently, these methods are more suitable to investigate the cluster of arbitrary shapes, they are more immune against the outliers and have good scalability (Berkhin, 2006) (Saraee *et al.*, 2007). Density-based clustering can be classified into density-based connectivity and density functions. The first algorithm in density-based connectivity is DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Han *et al.*, 2001). The good performance of DBSCAN depends on the good estimation of its input parameters. Therefore, a new approach called OPTICS (Ordering Points To Identify the Clustering Structure) develops the DBSCAN to compute the clustering parameters (Ankerst *et al.*, 1999). Finally, one of the most popular density functions methods is DENCLUE (DENsity based CLUstEring) (Hinneburg and Keim, 1998), which also influence by input parameters. ***Finally, it is clear that clustering according to the density needs to carefully identify some parameters basing on the characteristic of clustered objects, which require the predefining of these parameters or doing an additional computation to extract them. Also, this approach has been criticised for not being fully informative, and creating unbalanced clusters*** (Han *et al.*, 2001).

2.4.4. Grid-Based Methods

A grid-based approach built on ***space partitioning*** instead of ***objects partitioning***. ***Most Grid-based approaches provide a high quality of clustering (arbitrary shapes and resist to outliers) and relatively low complexity of computation O(N), but it does not serve this thesis because of its clustering, which is based on the density of objects rather than their connectivity. Also, this approach produces unbalanced clusters.***

Many algorithms are applying the grid-based approach; this section abstracts them as follows: The algorithm STING (STatistical INformation Grid-based method) suits the numerical attributes (spatial data) (Wang *et al.*, 1997). Also, WaveCluster method, which

has low complexity O(N), although its complexity exponentially increases with the rising size of the dimensions (Sheikholeslami *et al.*, 1998). Lastly, more complicated grid-based methods are developed to cluster high-dimensional objects, such as CLIQUE (CLustering in QUEst) (Agrawal *et al.*, 1998) and MAFIA (Merging of Adaptive Finite Intervals) (Goil *et al.*, 1999).

2.4.5. Graph Theory-Based Clustering

Graph theory-based clustering could be classified into ***graph-growing***, ***flows***, ***spectral-partitioning, recursive-graph-bisection, geometric-partitioning*** and ***gene-expression-data*** clustering methods (Buluç *et al.*, 2015). In addition to the previous six methods there are some other graph theory-based approaches categorised under different types such as Linkage Metrics (Xu and WunschII, 2005) and CHAMELEON (Karypis *et al.*, 1999) (see Section 2.4.2.,(A) Agglomerative (bottom-up)). Both Linkage Metrics and CHAMELEON are agglomerative clustering methods, but they use the distances between the graph nodes to describe the objects similarity (Xu and WunschII, 2005). The aforementioned six classes of graph theory-based clustering are summarised below:

A) Graph-Growing Methods: A graph-growing method using the ***BFS*** (Breadth-First Search) algorithm as a base for most of its versions (Buluç *et al.*, 2015). Then, the graph-growing method is developed to a newer version which called ***Bubble Framework*** (Walshaw *et al.*, 1995). Bubble Framework is an iterative method it splits the graph into more than two clusters (k > 2). The drawback of the ***Bubble Framework according to this work is that the beginning K nodes should be well distributed over the graph, while controllers' positions are constrained in the centre of graph***. In addition, it has a high complexity due to the iterative optimisation of the method.

B) Flows Method: Flows Method or ***HCS*** (Highly Connected Sub-graphs), uses the popular max-flow min-cut algorithm (Ford and Fulkerson, 2009) to divide the graph into unbalanced clusters, which have minimum cut edges (links) between them. This approach almost used as a subroutine inside another algorithm because it generates unbalance clusters (Buluç et al., 2015).

C) Spectral Partitioning Methods: The earliest model of the spectral partitioning method is ***spectral bisection***. It is optimised many times by (Donath and Hoffman, 1972), (Fiedler, 1975) and others. The ***spectral method is extended*** by Hendrickson

and Leland, (1995) to slice the graph into more than two clusters by applying multiple eigenvectors. ***However, this method does not require a high computational complexity to obtain the multiple eigenvectors; it can only partition the graph into four or eight clusters. The spectral clustering is used in many controller placement studies like (Xiao et al., 2014) (He et al., 2017)*** (see Sections 2.3.1 Minimizing network latency and 2.3.2 Maximizing resilience and reliability), ***and demonstrates the high computational complexity and unguaranteed balanced clusters.***

D) Recursive Graph bisection (RGB) Method: The **RGB** method uses the distance of shortest-path that connects the graph nodes instead of Euclidean distance (Simon, 1991). The algorithm needs to find the two furthest nodes for every cluster which require a high computational complexity, therefore, it uses some heuristic method to simplify this issue such as SPARSPAK (George and Liu, 1984). ***The drawback of the RGB method is that it can only create 2^k balanced clusters, where k=1,2,3...etc.***

E) Geometric-Partitioning Methods: In geometric-partitioning, the graph is divided into ***regions*** (clusters) according to the nodes (objects) ***coordinates***, which indicate the nodes similarity. The simplest method in this approach is Recursive Coordinate Bisection (***RCB***) (Buluç *et al.*, 2015)(Simon, 1991). There are several improvements to RCB such as ***inertial partitioning method*** (Williams, 1991) and the ***random spheres algorithm*** (Miller *et al.*, 1991). ***The weaknesses of RCB is producing skinny and long clusters. Also, it can only cluster the graph into 2^k balanced clusters*** (Simon, 1991).

A further, geometric-partitioning method is the ***Space-Filling Curves*** (Zumbusch, 2000) (Castro *et al.*, 2005). The Space-Filling Curves partitioning is a mixture of the grid-partitioning and the Space-Filling Curves techniques. ***The drawback of this method is determining the adequate density of grid-cells. Also, it is used to cluster the large-scale system with a proximate clustering and acceptable computation complexity but does not serve the present work which requires exact clustering. Finally, it does not consider the connectivity of the graph nodes which is the basic attribute in the wired network.***

F) Gene Expression Data Clustering: One of the algorithms which is established to group similar gene expression patterns in the cluster is CLICK (CLuster Identification via Connectivity Kernels) (Sharan and Shamir, 2000). ***The major disadvantages of CLICK are the fixed state of cluster threshold and the use of a heuristic technique which***

reduce clustering quality (Sharan and Shamir, 2000)***.*** Another graph theory algorithm developed to cluster gene expression data, is CAST (Cluster Affinity Search Technique) (Zhang, 2006) (Berrar *et al.*, 2003). ***The downsides of CAST are summarised by the needs for defining the affinity threshold (ŧ) and final clearing step*** (Bellaachia *et al.*, 2002).

2.4.6. Summary and Conclusion of Clustering Algorithms

The literature review of the clustering algorithm is written to simplify and summarise the investigation and research that has been done in the area of clustering algorithms, while the actual research includes more examinations and modifications in an attempt to employ the existing clustering methods to serve the proposed placement algorithm (see Section "5.8.2. The Design of the Peripheral Clustering Algorithm"). This review of the literature shows only an abstract of most common methods in every clustering direction and focuses on showing why the existing clustering algorithms are excluded. The conclusion summarises the literature review with adding some comparison tables for all methods. Initially, the review describes the requirements of the clustering approach which serves the proposed placement algorithm. Then, the main body of literature review displays the clustering methods as five classes (***partitioning***, ***hierarchical***, ***density-based***, ***grid-based*** (Lu, 2003) and ***graph-based*** (Xu and WunschII, 2005) ***methods***). On the one hand, the first four classes are reasonably studied, however they are not useful for the proposed placement algorithm due to its intention to cluster the objects according to their similarities with the difficulty of embedding the geometrical position and balancing the clusters. On the other hand, the fifth class is intensively investigated because it relates more to the network and supports the requirements of the proposed placement algorithm. However, it does not provide the complete solution. In addition to the pre-mentioned characteristics of the presented clustering methods, it is essential to present their computation complexity which plays an important role in nominating them to be utilised in a specific implementation. Therefore, the computation complexity of most common clustering algorithms is summarised in Table 2.7:

Clustering Method	Computational Complexity	References
Partitioning Relocation (K-Means)	$O(NKdI)$	(Xu and WunschII, 2005)
Partitioning Relocation (CLARA)	$O(K(40+K)^2+K(N-K))^+$	(Xu and WunschII, 2005)
Partitioning Relocation (CLARANS)	$O(KN^2)$	(Xu and WunschII, 2005)(Wang *et al.*, 1997)
Partitioning Relocation (EM)	Vary widely according implantation of E- and M-step.	(Dempster *et al.*, 1977)
Hierarchical clustering (Linkage)	$O(N^2)$	(Xu and WunschII, 2005)(Olson, 1995)
Hierarchical clustering (BRICH)	$O(N)$	(Xu and WunschII, 2005)
Hierarchical clustering (CURE)	$O(N_{sample}^2 . logN_{sample})$	(Xu and WunschII, 2005)
Hierarchical clustering (ROCK)	$O(N^2 + N\, m_m m_a + N^2\, logN)$ Where m_m: maximum number of neighbour. m_a: average number of neighbour.	(Guha *et al.*, 2000)
Hierarchical clustering (CHAMELEON)	$O(NK + N\, logN + K^2 logK)$	(Karypis *et al.*, 1999)
Density-Based (DBSCAN)	$O(N\, logN)$	(Xu and WunschII, 2005)
Density-Based (OPTICS)	$O(N\, logN)$	(Patwary *et al.*, 2013)
Density-Based (DENCLUE)	$O(N\, logN)$	(Xu and WunschII, 2005)
Grid-Based (STING)	$O(G_c)$ Where G_c:Grid cells	(Wang *et al.*, 1997)
Grid-Based (WaveCluster)	$O(N)$	(Xu and WunschII, 2005)
Grid-Based (CLIQUE)	Linear with number of objects, Quadratic with the number of dimensions	(Xu and WunschII, 2005)
Graph-Based (BUBBLE FRAMEWORK)	$O(KINE)$ Where O(NE): BFS computational complexity	(Buluç *et al.*, 2015)
Graph-Based (Spectral Clustering)	$O(N^3)$	(Liu *et al.*, 2013)
Graph-Based (Space-Filling Curves)	$O(N\, logN)$	(Luitjens *et al.*, 2006)
Graph-Based (CLICK)	$O(NE^{2/3})$	(Sharan and Shamir, 2000)
Graph-Based (CAST)	$O(N^2)$	(Bellaachia *et al.*, 2002)

General Notations:
N : Number of objects.
E : Edges (weight/ similarity) .
K : Number of clusters.
d : dimensions of attributes.
I : Iterations of process.

Table 2.7: Computational complexity of clustering methods.

Finally, From the pre-mentioned information, it is possible to summarise a general comparison of advantages and disadvantages of clustering methods as shown in Table 2.8.

Clustering Method	Advantages	Disadvantages
Partitioning Relocation	1) Easy to modify and implement.	1) Lacks stability and unbalanced clustering. 2) Identify a number of clusters before clustering (bad for dynamic clustering). 3) Weak against outlier and arbitrary shapes.
Hierarchical clustering	1) Does not need to identify the number of clusters before clustering. 2) Has dendrogram which represents the clustering hierarchy.	1) Difficult to select correct parameters which grantee the required clustering (find balanced clusters). 2) Almost has high computational complexity.
Density-Based	1) Does not need to identify the number of clusters. 2) Can discover arbitrary shapes. 3) Acceptable complexity.	1) Not accurate clustering and creating unbalanced clusters. 2) Quality of clustering depends on the identified threshold. 3) Not much informative therefore using an indexing technique (R*-tree).
Grid-Based	1) For clustering large dataset. 2) Can discover arbitrary shapes resist outliers. 3) Low complexity.	1) Not accurate clustering and creating unbalanced clusters. 2) Quality of clustering depends on the grid resolution.
Graph-Based	Most of them modified to support geometrical attributes and graph edges which are needed in the clustering requirements of the proposed placement algorithm.	Driven by different classes and has different characteristics.
Graph-Growing Methods (*Bubble Framework*)	1) Considering the nodes and links in clustering.	1) Beginning K nodes should be well distributed over the graph, while controllers positions consternate in the centre of graph.
Flows Method	1) Maximize reliability.	1) Used as subroutine inside another algorithm because it is generated unbalance clusters.
Spectral Partitioning Methods	1) Using the nodes connectivity to cluster the graph.	1) It can only partition the graph into four or eight clusters. 2) Cannot guarantee a balance clustering.
Recursive Graph bisection (RGB) Method	1) Using the nodes connectivity to cluster the graph.	1) Can only create 2^k balanced clusters, where k=1,2,3...etc.
Geometric-Partitioning Methods	1) Utilize the geometrical attribute of nodes in clustering.	1) Cannot produce the required clustering: A) RCB only 2^k clusters. B) Space-filling: Not accurate clustering. 2) Does not considering the connectivity of the graph nodes.
Gene Expression Data Clustering	1) The nodes (genes) similarities are represented as weighted edges.	1) Difficult to define the affinity threshold.

Table 2.8: Summary of the pros and cons of each type of clustering algorithms.

As demonstrated in the literature review of the clustering algorithm and previous tables there is no single clustering algorithm which could partition according to the geographical location and create balanced clusters of connected nodes in low computational complexity.

In addition to the review of clustering algorithms that are presented in section 3.3, the literature relates to the placement algorithms in section 3.2 showing some research that exploits many clustering algorithms to generate controllers domains, which are displayed in Table 2.9.

No.	Research	Clustering algorithm
1	(Zhang *et al.*, 2011)	Minicut clustering
2	(Xiao et al., 2014) (Xiao *et al.*, 2016)	Spectral clustering K-self-adaptive clustering
3	(Yao et al., 2014)	Capacitated K-center
4	(Aoki et al., 2015)	Minimum cut clustering Conductance clustering Distance-k cliques clustering
5	(Aoki and Shinomiya,2015)	Conductance clustering, Spectral clustering, Betweenness centrality clustering Repeated bisection clustering
6	(Liu et al., 2016)	Reliability Factor (K-Mean)
7	(Sanner et al., 2016)	Adapted the K-Mean Hierarchical clustering
8	(Zhu *et al.*, 2017)	Adapted K-Mean
9	(Zhao *et al.*, 2017)	Exemplar-clustering
10	*(Liao et al., 2017)*	Density-based clustering

Table 2.9: Summary of most clustering algorithms exploited for SD-WAN.

The studies presented in the table perform a different type of controller placement, however, their clustering algorithms encounter the problem of high computational complexity and cannot guarantee balanced clusters.

CHAPTER THREE
OPTIMISATION METRICS OF CONTROLLER PLACEMENT

3.1. Introduction

The previous research did not provide a view about the effect of the controller placement on the bandwidth utilisation. Also, they focus only on measuring the effect of the controller placement on the switch-controller latency and the flow-setup latency, which is just one portion (control-plane latency) of the first connection latency. According to the researcher, the optimisation ratio of the control-plane latency is not enough to determine if the achieved improvement really optimises the SD-WAN latency to a considerable level. Therefore, this research firstly examines the effect of the controller placement on the SD-WAN bandwidth using the old and proposed routing algorithm on the control packets. After that, it monitors the process of creating a full connection between two hosts in SD-WAN in order to: 1) construct the formula that contains all latencies, which are required when creating this connection; 2) measure the value of every latency component in this formula. The formula of the complete connection latency and the values of its components leads to determine the effect of the controller placement on the latency of creating a connection between two hosts in the SD-WAN.

This chapter demonstrates the mentioned objectives as three contributions, which are: the design of novel routing algorithm for SD-WAN; and the two new techniques to measure and evaluate the latency of the control plan and data plan of SD-WAN.

In the beginning, the chapter provides a complete review of the novel routing algorithm, by covering: its literature review; its design; its results; and its effect on SD-WAN bandwidth. Then, in a similar structure, the chapter explains the two different techniques, which used to measure the latency metrics of the SD-WAN. The first technique employs the computers (nodes in the data-plan) to measure the control plan latency metrics (all the components of the connection flow-setup latency) and the data plan latency metrics. While, the second technique employs the controller itself to measure the control path latency, which improves the validity of the previous

measurements of the first technique. Finally, performing this measurement is essential for determining the range of latency values of every latency metric. Also, it is the fundamental process for generating the formula, which is used to compute the latency metrics of the SD-WAN inside the COVN simulator.

3.2. Routing Algorithm Optimization for SD-WAN

Software Defined Network (SDN) provides a new fine-grained interface enabling the routing algorithm to have a global view of the network throughputs, connectivity and flows at the data-path (Adrichem *et al.*, 2014)(Jin *et al.*, 2013). Therefore, this research proposes a novel approach for a dynamic routing algorithm for SD-WAN; based on using a modified shortest-widest path algorithm with a fine-grained statistical method from the OpenFlow interface, called Shortest-Feasible OpenFlow Path (SFOP). This algorithm is designed to identify the optimal route from source to destination, and to provide efficient utilisation of the SD-WAN resources. It achieves this aim by considering both: the flow requirements; and the current state of the network. SFOP computes the optimal path, which provides the feasible bandwidth with the lowest hop count (delay). That will present better stability in SDN communication, QoS, and usage of available resources. Moreover, this algorithm will be the base for the SDN controller because it extracts the widest available bandwidth from source to destination for a single path. It enables the controller to decide whether it is enough to only use this simple algorithm, or if a more complicated algorithm that provides a larger bandwidth such as multiple-path algorithms is needed. Finally, the MATLAB Simulator computes the path using the SFOP algorithm. Then, the testbed is implemented using Pox controller, and Mininet emulator to apply the communication over the path of the SFOP algorithm. Comparing the latency of SFOP algorithm with the latencies of three other algorithm's (shortest, widest and shortest-widest routing algorithms), shows that this algorithm finds better latency for the optimal path. Evidence will be shown that demonstrates that SFOP has good stability in dynamic changes of SD-WAN.

3.2.1. The Motivation and the Objectives of SFOP Algorithm

In the traditional network, the ***routing algorithms*** find the best route from the source to the destination based only on the requirements of the flow. While, it is difficult for them to consider the status of the network to be optimised, when they are computing this route, for two reasons. First, this requires an extra communication to build and

update the database of the available resources and their utilisation (Adrichem et al., 2014). Second, it is difficult to provide the additional computation capability to execute the complicated algorithm on all the routers of the network (Trimintzios et al., 2000). SDN solved the above challenges and provided the motivation to develop the traditional routing algorithms by introducing the logically centralised controller with OpenFlow protocol, which allows it to control the data plan devices (switches). This new paradigm significantly reduces the communication overhead needed to have a global view of network resources utilisation. Also, SDN central control reduces the number of authorities, which request this information. Furthermore, the additional computation complexity is easier to provide in the logically central controller than on all network routers (Adrichem et al., 2014).

Finally, the objectives of the novel SFOP routing algorithm are as follows:

1. Finding the optimal path:
   a) It computes the shortest path of feasible bandwidth to have the path of the lowest latency for this bandwidth.
   b) It does not overuse the available bandwidth like widest and shortest-widest path routing algorithms (Ma and Steenkiste, 1998). Because it uses the feasible bandwidth as a base to search the shortest path.

   It does this without raising the time and space of the computation complexity to the limit, which could produce an unacceptable delay in the performance of the controller.
2. Optimizing the resources utilisation in real time because it uses OpenFlow statistic (nodes capacity and links bandwidth), which has a fast update and convergence for their database (Adrichem et al., 2014).
3. The SFOP algorithm will be the basis of the controller, which helps it to select whether it needs to implement a simple routing algorithm for the single path or will be required to apply a more complicated one for multiple-paths. It supplies the controller with a maximum bandwidth of the widest path. This data facilitates the controller to decide whether it needs to implement a more complicated routing algorithm such multi-flow algorithm (Márton Zubor, Attila Kőrösi, András Gulyás, 2014)(Aloul *et al.*, 2006).

3.2.2. The Related Works of the Routing Algorithm

History of routing algorithms for the traditional network has evolved from simple ones such as shortest and the widest path to the very complicated algorithms of constraint QoS (Márton Zubor, Attila Kőrösi, András Gulyás, 2014). Many works evaluate the performance of a routing algorithm and classify them according to their complexity and efficiency for a specific type of traffic (Ma and Steenkiste, 1998)(Steenkiste and Ma, n.d.)(SHESHADRI, 2004). The work in (Ma and Steenkiste, 1998) concluded that the shortest path algorithm functions fine for light traffic loads, while the widest-shortest path algorithm is convenient for heavy loads. The widest path algorithm performs badly for both heavy and light loads. While, the routing algorithm of shortest-widest path causes a low throughput for best-effort-traffic.

After that, a more complicated routing algorithm was developed to satisfy specific QoS requirements. Some authors provided an intensive review for the QoS routing algorithm (SHESHADRI, 2004)(Korkmaz and Krunz, 2003). Multi-Constrained Multipath (MCMP) routing algorithm was designed using a combination of multi-routing algorithms and redefined matrices [11]. The researchers in (Korkmaz and Krunz, 2003) developed an algorithm to find bandwidth and delay constrained path. Routing algorithms have attracted a great deal of research interest in the traditional network.

The emergence of SDN has motivated the reinvestigation of the routing algorithm due to the new features of a novel paradigm.

On the one hand, several works are implemented using a simple algorithm such as the shortest and widest routing algorithm to optimise routing in SDN. For example, in (Ingale and Kakade, 2014) the widest or shortest path was used, after the application server notifies the controller, which one is more suitable for the current flow. Also, in one of the recent works in this area (Yalda *et al.*, 2015), they replaced the shortest path with the widest path algorithm in a Floodlight controller. As noted, this work did not develop many new aspects of the simple polynomial routing algorithm, which could exploit the new features of SDN or to satisfy its new requirements. They mostly implement and evaluate the traditional algorithms in SDN network. Therefore, this research aims to fill this gap.

On the other hand, most of the research that optimised the routing algorithms in SDN focused on the complicated routing algorithms, which are destined to serve many

constraining QoS specifications. For instance, (Egilmez *et al.*, 2011)(Egilmez *et al.*, 2012) used very complicated algorithms to optimise QoS for multimedia and video streaming. Similarly (Egilmez *et al.*, 2013), constructed a multi-constrained shortest path to enhance video streaming. Subsequently, (Nakibly *et al.*, 2012) worked to get a better QoS with a minimal controller load at the same time. After that, (Aloul *et al.*, 2006) utilised the advanced Boolean satisfiability (SAT) techniques to compute the constrained shortest path. Some researchers in (Al-Jawad *et al.*, 2015) have employed Bayes' probability theorem to find the feasible link of satisfied QoS constraint. Then, *K-maximally* disjoint path algorithm was presented by (Abe *et al.*, 2015). This work also aimed to provide a buffer, which prevents the implementation of these complicated algorithms until the flow requirements enforce the controller to implement them. Eventually, a new piece of research developed a new routing protocol which aims to reduce the power consumption in SDN data center, as well as it reduces the failure rate in its devices (Al Mhdawi and Al-Raweshidy, 2018).

After the above research, this work develops SFOP algorithm, which provides a path of lower latency and feasible bandwidth. At the same time, it keeps acceptable complexity according to the SDN controller capabilities. In addition, it focuses on replacing the informative old protocol, and the complex computation requirements to update the resources utilisation; by a fine-grained OpenFlow interface, which is one of the main keys to the success of SDN.

Finally, it is clear that the shortest-widest path algorithm negatively consumes the network bandwidth, as mentioned in the first paragraph. Therefore, it is modified to search the shortest path of feasible bandwidth for the flow.

3.2.3. Statistics of OpenFlow Interface

This work suggests using the fine-grained statistics of OpenFlow interfaces to computing the optimal path by routing algorithms. Especially, the information required for a routing algorithm is extracted by efficient open-source software's, such as Open Traffic matrix (Open TM) (Tootoonchian *et al.*, 2010) and Open Network Monitoring (OpenNetMon) from the OpenFlow interface (Adrichem *et al.*, 2014).

This section briefly shows OpenFlow protocol and the related routing algorithms requirements. OpenFlow protocol is designed to have a global view of all network connectivity and throughput for dataflow layer. When any change in network

connectivity happens, OpenFlow switches send a ***port-state message*** to the controller to update this change (Open Network Foundation, 2009)**.** In addition, OpenFlow switches have flow counters ***per-port***, ***per-flow***, and ***per-queue***. These counters maintain the network state in the controller database using two types of messages. The more frequent one is the ***flow-removed*** message when the flow is time up. The second message is ***Read-State***, which is sent by the controller to collect statistics from switches. Both messages contain the ***flow-duration*** and its ***byte count*** at each switch, which enables the network throughput for each link to be identified (Open Network Foundation, 2009)(Open Networking Foundation, 2013).

The developed routing algorithm needs to learn the network throughput as an incident matrix of bandwidth (***IncBW***) (Adrichem *et al.*, 2014)(Tootoonchian *et al.*, 2010). In addition, this software should supply the algorithm by the feasible bandwidth per flow type (***FBW***) based on monitoring and the QoS requirements (Jin *et al.*, 2013)(Adrichem *et al.*, 2014).

The algorithm also uses the OpenFlow to identify the type of flow based on different parameters in OpenFlow header, which are source port (***TCP/UDP-src-port***), destination port (***TCP/UDP-src-port***) and type of service (***IP-TOS***) (Open Network Foundation, 2009).

3.2.4. The Design of the SFOP Routing Algorithm

It is based on a combination of shortest widest path and the restricted shortest path. The algorithm is executed as a management application inside the SDN controller. The algorithm has two input parameters, which are the incident matrix of residual bandwidth (**IncBW**) in the current state of the network and the feasible bandwidth (**FBW**) for the current flow. Both are learned from the open source software presented in (Adrichem *et al.*, 2014)(Jin *et al.*, 2013), which use the fine-grained OpenFlow statistic as explained in Section 3.2.3.

The algorithm works as follows: First, the algorithm computes the widest path and widest bandwidth; Second, it compares the FBW of OpenFlow statistics with the found the widest bandwidth; Then, it uses the FBW if it is less or equal to the widest bandwidth. Otherwise, it uses the widest bandwidth itself; Third, it finds the shortest path of the chosen bandwidth. The second and third parts (the modified parts) are the contributions of this paper. Unlike the algorithm of the shortest widest path, that finds

the shortest path of widest bandwidth. The SFOP algorithm finds the shortest path of feasible bandwidth.

The algorithm computes the optimal path, which provides the feasible bandwidth and the minimum hops count (minimum delay). In the meantime, the algorithm also optimises the utilisation of links bandwidth, because the algorithm will always use the path, which has the feasible bandwidth in the current state of SD-WAN and leaves the links, which have the widest bandwidth for best-effort-traffic.

Let's formulate the algorithm for a more detailed demonstration. The SD-WAN is represented by an undirected finite graph $\boldsymbol{G}\ (N, L)$, where the $\boldsymbol{N}$ is the network nodes and $\boldsymbol{L}$ is the links between them. $|N| = n$ and $|L| = l$ .The path $\boldsymbol{P}$ is the optimal path of sequence nodes $\boldsymbol{P} = (\boldsymbol{In} = \boldsymbol{n1}; \boldsymbol{n2}, \dots, \boldsymbol{nk} = \boldsymbol{Eg})$, where $\boldsymbol{In}$ is Ingress node, $\boldsymbol{Eg}$ is Egress node, $\boldsymbol{k}$ is the length of the path and $(n_i; n_{i+1}) \in L : \forall i = (1, \dots., k-1)$. A link $\boldsymbol{l}$ with origin node $\boldsymbol{n}$ and destination node $\boldsymbol{m}$ is denoted by ($\boldsymbol{n}$,$\boldsymbol{m}$). Respectively, with each link $\boldsymbol{l} = (\boldsymbol{n}, \boldsymbol{m}), where\ \boldsymbol{n}, \boldsymbol{m} \in \boldsymbol{N}$ there is an associated bandwidth $\boldsymbol{BW_l \geq 0}$ and a delay $\boldsymbol{\delta_l \geq 0}$.

The main steps of SFOP routing algorithm:

1. Find the less utilized links, which connect every neighbour of nodes (create the Adjacency matrix **AdjBW** from incident matrix **IncBW**. See pseudocode in Figure 3.1.
2. Compute the widest path using modified Dijkstra's algorithm. See Figure 3.2.

   The widest path is defined as shown in Equation 3.1:

$$\boldsymbol{Bandwidth(widest_P)} = \boldsymbol{Max}_{\boldsymbol{l_i=(l_1,l_2,l_3 \dots l_I:\ l_i \in Adl\ and\ I \leq ADC)}} [\mathbf{min}(\boldsymbol{BW_{li}}, \boldsymbol{D_i})] \quad (\mathbf{3.1})$$

   Where:

   a) *D = Distance matrix, which saves the maximum bandwidth of path from the ingress node until this node.*
   b) *Adl = the group of links connected to adjacent nodes.*
   c) *Adc = the count of these links.*

3. Check bandwidth; the algorithm uses the feasible bandwidth (FBW) if it is less than the widest bandwidth ($widest(BW_P)$). Otherwise the widest bandwidth ($widest(BW_P)$) is used. See Figure 3.3.
4. Remove all the links, which have a bandwidth less than the last specified bandwidth ($widest(BW_P)$ or FBW) . See Figure 3.3, step 31.

5. Compute the shortest path of last specified bandwidth, using another modified Dijkstra's algorithm. See Figure 3.4.

The cost of shortest path is defined in Equation 3.2:

$$\boldsymbol{Cost(shortest_P)} = \sum_{k=0}^{N} \underset{l_i=(l_1,l_2,l_3 \ldots l_I:\ l_i \in Adl \text{ and } I \leq Adc)}{\mathbf{Min}} [\, \boldsymbol{BW_{li} + D_i}] \qquad (3.2)$$

Where:

a) D = Distance matrix, which saves the accumulated bandwidth of path from the ingress node until this node.
b) Adl = the group of links connected to adjacent nodes
c) Adc = the count of these links.

Initialize 1# create **Adjacency matrix of bandwidth (AdjBW)** by extracting it from Incident matrix of the graph (**IncBW**).
(G: GRAPH, NODE= N, Links=L, Incident matrix of bandwidth=**IncBW**, Adjacency matrix of bandwidth = **AdjBW**)
1. Input: incident matrix of bandwidth
2. **For** $all\ n \in N$ **do** . /* *for all nodes in G* *|
3. chose the link of higher bandwidth; |* *to find the less utilized link if there are more than one link* *|
4. **Endfor**

Figure 3.1: The pseudo code for creating adjacency matrix of bandwidth.

Algorithm1# compute the widest path##########################.
(Ingress node=**In**, Egress node=**Eg,** Visited nodes vector=$\boldsymbol{V}$, Unvisited node vector=$\boldsymbol{UV}$, Distance matrix=**D**, Current Visit Node **=CVN,** path=**P,** Path bottleneck bandwidth= $\boldsymbol{widestBW}$).
Compute distance matrix
5. **Initialize**: $V = \emptyset$; |* *initialize visited node vector to be empty**|
6. **Initialize**: $UV = all\ N$; |* *initialize Unvisited node vector to have all N**|
7. **Initialize**: k=0; |* *initialize counter to zero**|
8. **Initialize**: $\boldsymbol{CVN = 0}$;|* *variable to hold the Current Visit Node (CVN)* *|
9. **While** $UV \neq \emptyset$ **do**|* *for all nodes in UV* *|
10. **k=k+1**; |* *increment counter**|
11. $\boldsymbol{CVN =\ maxdistance}(n)$ |*Select node n has maximum distance D to be current visited node**|
12. $\boldsymbol{Insert}\ CVN\ \boldsymbol{in}\ V$
13. $\boldsymbol{Remove}\ CVN\ \boldsymbol{from}\ UV$
14. **For** all neighbours $[CVN]$ **do**
15. $\boldsymbol{D} = [\boldsymbol{min}(\boldsymbol{AdjBW_{lk}}, \boldsymbol{D_{k-1}})]$
|* *update the bottleneck bandwidth in D for all neighbours of CVN whose yet unvisited* *|
16. **Endfor**
17. **EndWhile**
Compute widest path and its bandwidth from distance matrix.
18 **Initialize**: P= Eg; |* *initialize path vector (P) to equal Egress node**|
19. **Initialize**: temp-node=Eg; |* *initialize variable to hold last node in path**|
20. **Initialize**:$widestBW = \infty$; |* *initialize variable to hold the bottleneck bandwidth of widest path**|
21. **While** $temp - node \neq In$ **do**
22. $temp - node = last\ node\ of\ P$
23. $\boldsymbol{Insert}\ \ \boldsymbol{n}\left(max\left(\ D\ of\ all\ neighbours_{\boldsymbol{temp-node}}\right)\right)\ to\ \boldsymbol{P}$ |* *Update path vector**|
24. $\boldsymbol{widestBW} = min(\boldsymbol{previous}\ widestBW, \max D)$
25. **Endwhile**

Figure 3.2:The pseudo code of algorithm 1 (computing the widest path).

Initialize2# Find and Use Best Available Bandwidth ########.
(Feasible bandwidth of flow =**FBW**, Best Available bandwidth = ***BABW*).**
Best Available bandwidth (BABW) equal to the feasible or widest bandwidth.
Input: FBW; */*input the Feasible bandwidth (**FBW**) for this flow */*
Initialize: $\boldsymbol{BABW} = 0$;
*/*Variable hold the value of Best Available bandwidth (BABW) */*
If $FBW \geq widestBW$ **then** $\boldsymbol{BABW} = FBW$
*/*if the feasible bandwidth is available then use it*/*
Else $\boldsymbol{BABW} = widestBW$
Endif
###Remove any bandwidth lower than BABW from adjacency matrix of bandwidth (AdjBW).
For $all\ n \in N$ **do** */* for all nodes in G */*
If $BW_n < BABW$ **then** $BW_n = 0$; */* set unwanted bandwidth to zero*/*
Endfor

Figure 3.3: The pseudo code for finding and using the best available bandwidth.

Algorithm 2#compute the shortest path##############################.
###prepare adjacency matrix of bandwidth (**Adj*BW***).
For $all\ n \in N$ **do** |* *for all nodes in G* *|
If $AdjBW_n \leq 0$ **then** $AdjBW_n = \infty$;
|* *set zero bandwidth to infinity**|
Endfor
updates node distance and path.
Initialize: $V = \emptyset$;|* *initialize visited node vector to be empty* *|
Initialize: $UV = all\ N$;
|* *initialize unvisited node vector to have all nodes N**|
Initialize: k=0 |* *initialize counter to zero**|
Initialize: $CVN = 0$
|* *it is a temporary variable to hold the Current Visit Node (CVN)* *|
While $UV \neq \emptyset$ **do** |* *for all nodes in UV* *|
k=k+1; |* *increment counter**|
$\boldsymbol{CVN = mindistance(n)}$;|**Select n with minimum distance D* *|
$\boldsymbol{Insert}$ *CVN* $\boldsymbol{in}$ *V;*
$\boldsymbol{Remove}$ *CVN* $\boldsymbol{from}$ *UV*
For all neighbours [$chosen$] **do**
$\boldsymbol{D = [AdjBW_{lk} + D_{k-1}]}$ |* *update the bottleneck bandwidth in distance matrix D for all all neighbours of CVN whose yet unvisited* *|
$\boldsymbol{P = n(min(\ D\ of\ all\ neighbours_{temp-node}))}$
|**Update shortest path**|
Endfor
EndWhile

Figure 3.4: The pseudo code of Algorithm 2 (compute the shortest path).

3.2.5. Analysing the Computational Complexity of SFOP Routing Algorithm

As have been shown in the previous section the SFOP algorithm is composed of two main parts. Both parts are based on modified Dijkstra's algorithm (Barbehenn, 1998). Algorithm 1, shown in Figure 3.2, is responsible for computing the widest path. The worst-case complexity of Algorithm 1 is $\boldsymbol{O(NlogN + L)}$ (Kaibel and Peinhardt, 2006),

where $\boldsymbol{N}$ is network nodes, and $\boldsymbol{L}$ is network links. Similarly, Algorithm 2's computational complexity is $\boldsymbol{O}(\boldsymbol{NlogN} + \boldsymbol{L})$(Barbehenn, 1998)**,** where $\boldsymbol{N}$ is the number of network nodes, and $\boldsymbol{L}$ is network links. Therefore, the worst-case computational complexity for all algorithms is $\boldsymbol{O}(\boldsymbol{NlogN} + \boldsymbol{L})$.

From a simple comparison of computational complexity, between this algorithm and other polynomial routing algorithms, has been presented in (Orlin, 2013)**.** It has been observed that most of the polynomial algorithms used to find the optimal path have a similar time complexity of our algorithm. However, SFOP algorithm still reduces time complexity because in logically centralise control of SDN it is executed fewer times in comparison to distributed control of the traditional network.

3.2.6. SFOP Emulation Environment

This section introduces the hardware and software tools used for the emulation. A computer with a corei7 of the 2.4GH processor and 16GB memory is used to implement the emulation. The Matlab programming language is used to write the management routing algorithms. This program reads the incident matrix of bandwidth (IncBW) and feasible bandwidth of current flow (FBW). After that, it computes the optimal path (P) using the SFOP algorithm.

Next, a Python application program is developed for the Pox controller to read the optimal path and install the required rules to OpenFlow switches. In addition, it updates the bandwidth matrix of SD-WAN. Pox controller is used due to it is easy to program and has no faults (Kreutz *et al.*, 2015b).

Another, Python program is written to create the SD-WAN topology. Finally, Mininet emulator is used to emulating the network of OpenFlow virtual switches (OVS) and execute the rules Pox controller.

3.2.7. Implementation and the Results of the SFOP Algorithm

A virtual network is generated to emulate SD-WAN. This network is designed to have similar characteristic to the real WAN topology, as shown in Figure 3.5.

The topology has one Pox controller and sixteen OVS switches, which is the maximum port capacity for a Pox controller. The bandwidth of the backbone links is 1 Gbit, while the forked links have 100 Mbit and 10 Mbit bandwidth.

A Pox controller is placed in the position of node eleven, because it is located in the centre of SD-WAN and lays on its backbone infrastructure.

Two different tests are implemented in this algorithm. The first one is to compare the path latency of SFOP algorithm with the path latency of shortest, widest and shortest-widest algorithms to specify which algorithm has the best performance. The second comparison is for path latency of SFOP algorithm in multiple states of the network to evaluate the algorithm performance with dynamic changes.

Both tests are applied to three sizes of packets (1 KB, 10 KB, and 64 KB) to evaluate the performance of the routing algorithm with different loads. Moreover, both use the single ingress node and Egress node.

A) Test one: shows the different paths of the routing algorithms as shown in Figure 3.5. Route one shows the shortest path. It also represents SFOP route when small packets are sent (feasible bandwidth required is less than 10 Mbits). Route two displays the shortest path. In addition, it presents SFOP route, when the packet size is larger than 10 KB (a feasible bandwidth larger than 10 Mbits is specified). Finally, route three expresses the widest path.

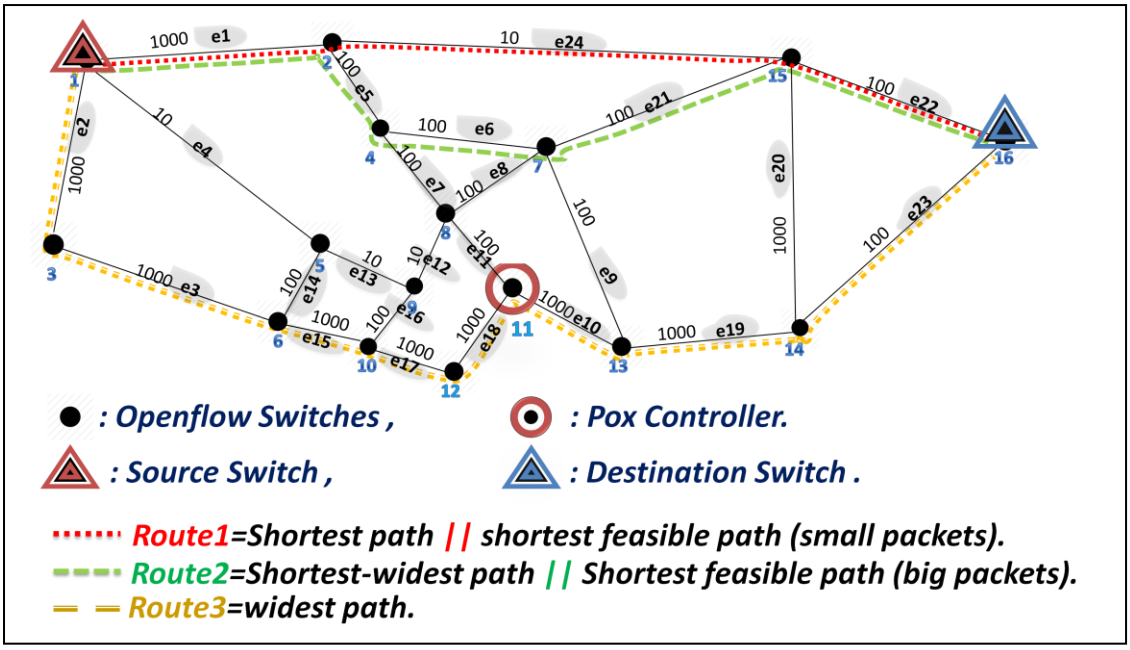


Figure 3.5: The paths of the different routing algorithms in SD-WAN (Note: 10,100 and 1000 represent the link bandwidth in Mbit).

Consequently, the emulation generates the latency results, which are displayed in Figure 3.6.

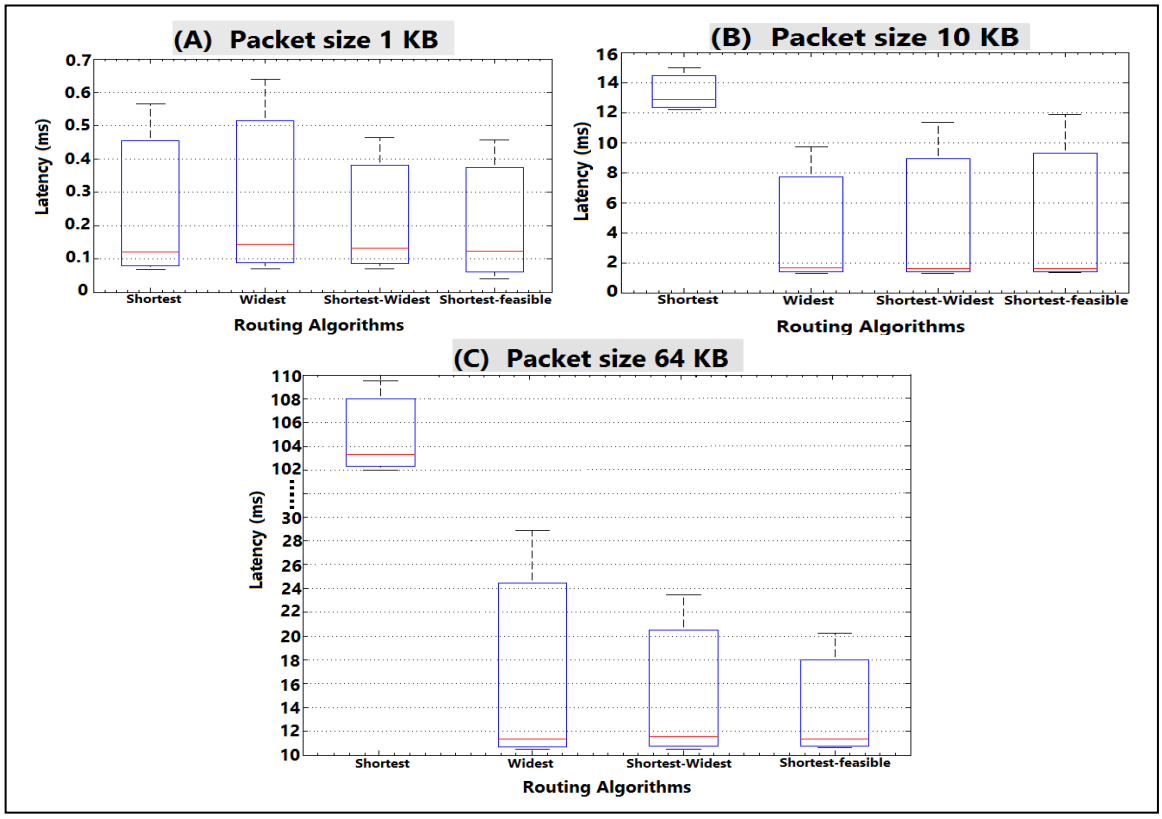


Figure 3.6: The latencies for different routing algorithms.

These results show that the shortest path and SFOP algorithms have the best mean of latency for small packets. While, for the big packets the shortest path algorithm, is very bad, and the other three have similar results. Which means that the shortest-feasible bandwidth does better than the others in all loads. See the comparison summary in Table 3.1.

No.	Packet size	Shortest (ms)	Widest (ms)	shortest-widest (ms)	shortest-feasible (ms)
1	1 KB	Best (0.108)	Good (0.147)	Better (0.125)	Best (0.111)
2	10 KB	Bad (12.894)	Best (0.145)	Best (1.607)	Best (1.581)
3	64 KB	Bad (103.345)	Best (11.342)	Best (11.517)	Best (11.357)
Notes:				Consume residual Bandwidth	Not consume residual Bandwidth

Table 3.1: Latency performance of different routing algorithms.

B) Test two: it shows the different paths of SFOP algorithm when the network links have a different residual bandwidth (IncBW), and the feasible bandwidth (FBW) is fixed to be larger than 100 Mbit. See Figure 3.7.

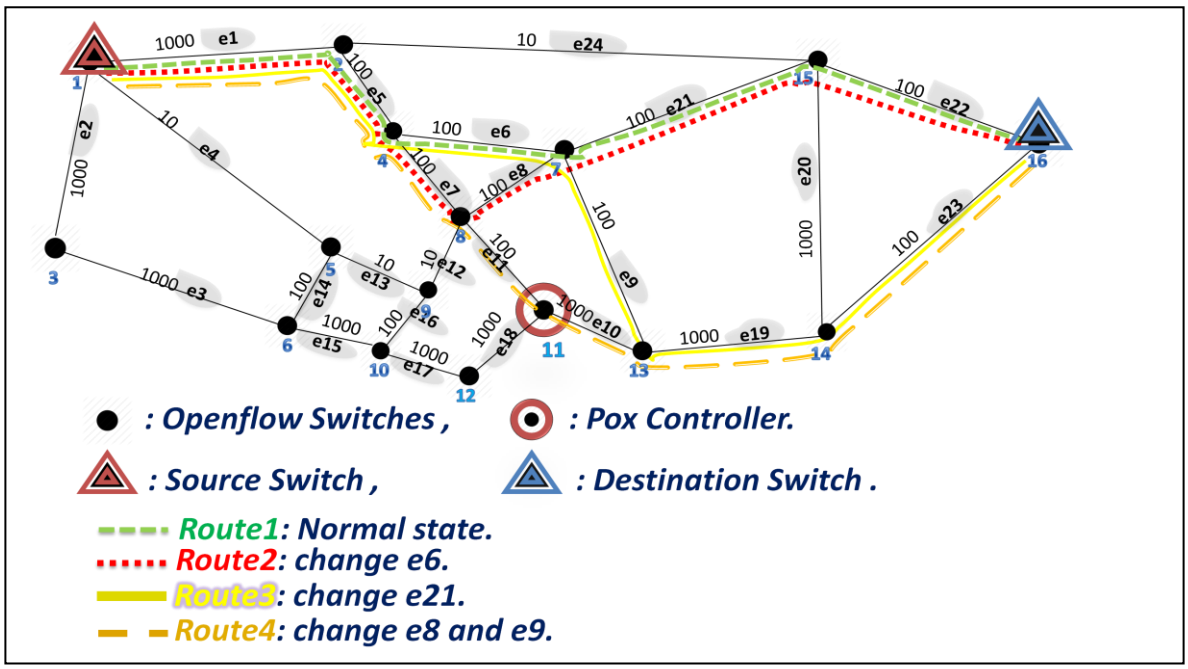


Figure 3.7: Several paths of SFOP algorithm with the dynamic changes of the bandwidths.

Route one shows the path in a normal state (full bandwidths). Route two displays the path if the bandwidth of link **e6** down from 100 Mbit to 10 Mbit. The third route presents the path in case link ***e*21** goes from 100 Mbit to 10 Mbit. Finally, route four, represent the path when the links ***e*8** and ***e*9** have residual bandwidths 10 Mbit instead of 100 Mbit in the initial state.

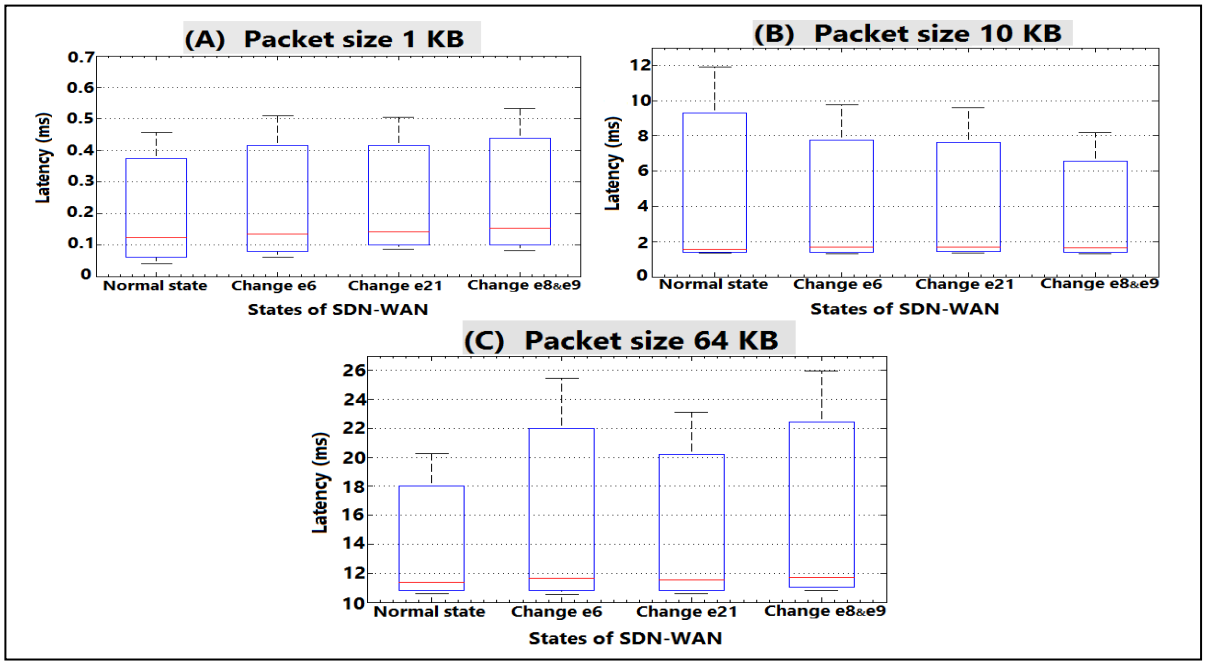


Figure 3.8: The latency of the dynamic routing in different states of SD-WAN.

As a result, the algorithm shows a similar latency for all cases, if an alternative path of similar bandwidth is available, See Figure 3.8. In summary, the algorithm provides good stability for flow path in SD-WAN.

Bearing in mind that the test is done using ping command with different bandwidths to examine the algorithm latency, which is the focus of this research.

3.2.8. The Conclusion of the SFOP Algorithm

In this section, I proposed a routing algorithm called SFOP to find the optimal path and to enhance the network resources utilisation, in SD-WAN. The emulation of the algorithm in Mininent, shows that it provides better latency than other comparable polynomial routing algorithms (shortest, widest and shortest-widest path). It also shows it has good stability with dynamic changes of the network in most cases. Moreover, SFOP provides valuable data for the controller to build its decision to apply the further complicated algorithm as demonstrated in Section 3.2.4. Finally, it provides the resources utilisation of SD-WAN without increasing the computation complexity. It is noteworthy that, this algorithm does not produce a different result from the shortest path routing algorithm, if it is implemented for the control-packets between the controller and its switches, because they are a small packet. This finding is obtained only after completing the tests of routing algorithms and lead the research to keep using the shortest path algorithm for control packets.

3.3. Developed Asynchronous Technique to Evaluate the Performance of the SDN Switches Using the Hosts in Data-Plane

This section presents the second contribution, which objecting to measure the values of latency metrics, which are required for creating a connection between two hosts in SD-WAN. Besides that, it is extracting the other performance metrics, such as bandwidth, packet losses and jitter of the SDN physical equipments. Also, it is objecting to producing the latency equation (formula), which is needed to compute the connection flow-setup latency and its components in this research (see Section 3.3.5).

A new technique is developed by this research to accurately measure the data-plan and control-plane latency metrics using the host nodes in the data-plane. The conducted tests provide realistic performance metrics, which could be used in the research computations. Also, this research addresses the differences between the SDN virtual environment and physical SDN switches in order to modify the SDN virtual environment to make it more realistic. Consequently, this research presents a precise performance evaluation and comparison of off-the-shelf SDN devices, HP Aruba 3810M, with Open

Virtual Switch (OVS) inside Mininet emulator. This work examines the variability of the path delay, throughput, packet losses and jitter of SDN in a different windows size of the packets and network background loads. The conducted experiments consider a number of protocols such as ICMP, TCP and UDP. The developed technique shows more precise results in comparison to other techniques. The comparison of results shows a dissimilarity in the behaviour of SDN hardware and the Mininet emulator. The SDN hardware exposed higher latency and connection flow-setup time due to extra resources of delay, which the emulator does not possess. Finally, the results of latency metrics demonstrate that the control-path latency is much smaller than other latencies of creating the connection in SD-WAN.

3.3.1. The Related Works of the Measurement Techniques

Both the physical and emulated SDN test-beds need to be evaluated, which demands the unified and idealistic measurement tools for both test-beds. Although, there has been a significant amount of research in the field of network measurement, it is imperative to: (1) Identify and adjust a suitable measurement technique which serves the aim of this evaluation; (2) Investigate the findings of other research; (3) Exploit their results as additional evidence that verifies the accuracy of the result this research. Therefore, a comprehensive literature review of network measurement is performed to avail the purposes above. The related work is classified into three parts: (A) Traditional measurement techniques and matrices; (B) Evaluation of virtual switches and emulator; and (C) Evaluation of SDN.

A) Traditional Measurement Techniques and Matrices

In traditional networks, two-way latency can be obtained using: (a) Ping (ICMP) (Wang *et al.*, 2004); (b) (SYN-ACK/ACK) packets of TCP three handshake (Dhaval N. Shah *et al.*, 2002); (c) TCP Timestamp option (V. Jacobson *et al.*, 1992); and (d) Two-way UDP packets with embedded timestamp of connected hosts (Wang *et al.*, 2004). Ping is a powerful tool to extract the end-to-end Round-Trip-Time (RTT). It is a very effective tool to measure the latency variance in the different windows size of packets. However, it gives a rough estimation for one-way latency (Wang *et al.*, 2004). Ping tools predominate in latency measurement research as could be notified in (Prieto *et al.*, 2015)(Wang *et al.*, 2004). The second technique to calculate the RTT is based on the difference in timestamp between the SYN-ACK packet and ACK packet (second and third

packets) of the TCP connection establishment on the callee side (Dhaval N. Shah *et al.*, 2002). (SYN-ACK/ACK) technique is unlike ping in the ability to change the window size of packets. It can find the RTT only for the small window size of packets (the packets of TCP three handshake). Both techniques suffer the additional latency which resulted from the processing time in the second end-host (Prieto *et al.*, 2015). The third method to obtain the RTT is enabling the TCP timestamp option in TCP header (V. Jacobson *et al.*, 1992). It provides an acceptable precision of transmission latency, but it produces an extra payload added to the real load of original traffic. Sometimes, the packet with enabled TCP timestamp is terminated by firewall devices because it reveals the timestamp of communicated nodes, which is clearly described in (Prieto *et al.*, 2015). Finally, using the two-way UDP with an embedded timestamp of connected hosts to detect the Round-trip Delay Time (RTD) or One-way Delay (OWD). Like the TCP timestamp option, it supplies an accurate latency, but it requires a customised benchmark which can test only the latency of UDP packets (Sans and Gamess, 2013).

B) Evaluation of Virtual Switches and Emulator

Open vSwitch was used to emulate the SDN network in Mininet emulator, because it can provide complete functionality of the OpenFlow switches (Mininet Team, 2016). Several works evaluate Mininet for different purposes using the common measurement tools. The works in (Keti and Askar, 2015)(Danielis *et al.*, 2015)(Gustaf Jan Gunnar Nilstadius, 2016)(Wang, 2014) measured the RTT of Mininet using ping utility. Others evaluated the RTT with ping and bandwidth of links using Iperf tools (Koerner and Kao, 2014)(Zoher Bholebawa and Dalal, 2016). Finally, the ping ICMP packets were used to indicate controller flow setup time, while the TCP, UDP packets were emulated to estimate application latency in the Mininet (Turull *et al.*, 2014). All pre-mentioned works missed defining the bandwidth limit of links inside Mininet, which created an unrealistic measurement. However, their results were a good starting point to realise the behaviour of SDN network and Mininet emulator.

C) Evaluation of SDN

The logically centralised control of SDN adds new measurement fields such as the latency of the control layer. In SDN, the methods of determining the network latency would be handled by the data plane or by the control plane.

On the one hand, most works used the controller to compute the path latency in different ways. The authors in (Phemius and Bouet, 2013)(van Adrichem *et al.*, 2014)(Yu *et al.*, 2015) sent probe packets from the controller, which passed through the path back to the controller. The path latency is extracted by subtracting the control path latency from the difference of sending and receiving timestamps. The work in (Shuo Wang *et al.*, 2017)(Agarwal *et al.*, 2014)(sFlow.org, 2012) uses the same method except for the way of determining the path. The path is passively determined after collecting all the entries of the flow-table from the OpenFlow switches, then sending a probe packet from the controller to compute the paths latency. Another piece of research uses the looping technique by applying a loop of special service packets through the path with specified Time-to-live (TTL). OpenFlow switches on that path: (a) decrement TTL; (b) register the number of iterations; (c) and forward the packet in the loop while the TTL is not zero, otherwise, forward it to the controller. The controller then calculates the latency using TTL and iteration number (Sinha *et al.*, 2015)(Altukhov and Chemeritskiy, 2014). Another piece of research proposes the Queue Length Method (Sinha *et al.*, 2015). It considers the processing, propagation and transmission delay as constant values and uses the detected queuing delay to estimate the path delay.

On the other hand, some researchers deny over-heading the controller with additional computation. Also, they noticed that the control layer latency varies more than the switch forwarding latency, which heavily affects the accuracy of latency measurement of the data path (Shuo Wang *et al.*, 2017)(Atary and Bremler-Barr, 2016). For example, (Atary and Bremler-Barr, 2016) employs a monitoring host in the data-plane to measure the path latency using an active probe packet. Also, (Koerner and Kao, 2014) uses the traditional ping and Iperf tool to identify the latency and bandwidth of the network in his study.

Finally, a few studies focus on resolving the control layer latencies. Kuniar *et al.*, (2014) create a python script to probe packets between OpenFlow switches and the controller to dissect fractions of the control layer latency. Sonchack *et al.*, (2016) study the effect of varying the load of the control plane on its latency using background control probe packets and ping utility on the Pica8 physical switch. Lastly, High-Fidelity Switch Models for SDN Emulation (Huang *et al.*, 2013) is the work most related to this paper. It tests and compares the flow-setup latency of the HP ProCurve, Quanta, and Monaco physical switches with OVS. The OVS was emulated on Linux based virtual machine and not in

Mininet. The test in (Huang *et al.*, 2013) is limited to measure the control layer latency under different conditions while missing the measurement and comparison of the data-plane latency, While this research measures the data plane latency on the physical and Mininet testbeds.

3.3.2. Description of the Proposed Asynchronous Latency Measurement Technique

Since the intent here is to extract actual path latency and not its approximation, ping utility might not fulfil the purpose by itself. For the reason that, the RTT which is computed by the ping tool comprising an extra processing time which belongs to the Callee side. Therefore, simple steps were added to optimise the latency which was obtained from RTT of the ping tool, as follows:

1. Ping from sender to receiver as shown in Figure 3.9.

2. Capturing the send request at both sides to record the timestamp of:
   a) Send-Request-Time (SReqT) at the sender.
   b) Received-Request-Time (RReqT) at receiver.

3. Capturing the returned reply at both sides to record the timestamp of:
   a) Send-Reply-Time (SRepT) at receiver.
   b) Received- Reply -Time (RRepT) at the sender.

4. Compute processing time (Δ2) in the receiver side "callee side" by subtracting (SRepT) from (RReqT):

$$\Delta 2 = (SRepT) - (RReqT) \quad (3.3)$$

5. Compute the actual Latency from ping RTT in two steps:
   a) Calculate the ping RTT (Δ1):

$$RTT = \Delta 1 = (RRepT) - (SReqT) \quad (3.4)$$

   b) Calculate two-way-latency (Δ3):

$$Two\text{-}way\text{-}Path\text{-}latency = \Delta 3 = \Delta 1 - \Delta 2 \quad (3.5)$$

6. Compute the one-way-latency:

$$One\text{-}way\text{-}latency = \Delta 3/2 \quad (3.6)$$

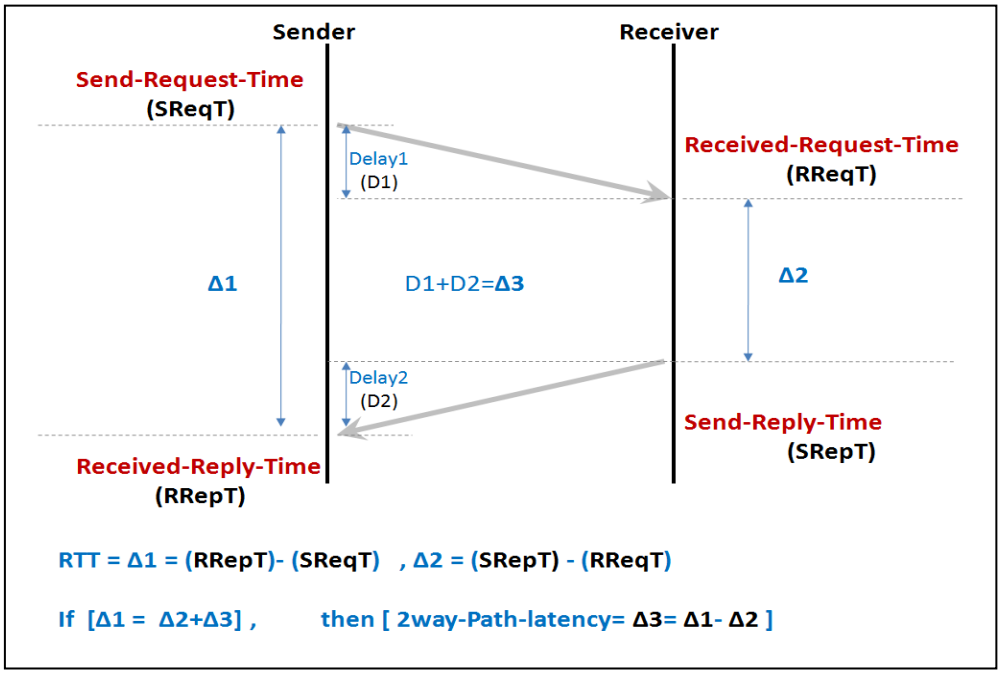


Figure 3.9: The proposed asynchronous latency measurement technique.

The (Δ3) represents two-way-latency through the switch with considering that the link delay is negligible. For the reason that the maximum propagation delay for 1 meter is approximate to 5.5 ns/m (Beckhoff Automation GmbH & Co. KG, 2017). One-way latency is half of two-way latency, because SDN provides the same path for the request and the reply packets. In that technique, it is possible to compute the two-way or one-way path latency accurately. Also, it demonstrates parity with one-way synchronised techniques without the severing of synchronising the end host or customising a software benchmark. Furthermore, it can be used to measure the accurate latency for any traffic such as TCP, UDP, ICMP, etc.

3.3.3. Testbeds Description and Methods

This section briefly describes the testbed components and their topologies, followed by the measurement methods.

A) Testbeds description:

The tests are performed in the SDN laboratory of the University of Northampton. Two testbeds were used to examine the physical and emulated SDN. The physical testbed comprises of HP VAN SDN Controller 2.7.18 (Packard Enterprise, 2016a) which runs over a VirtualBox on a dedicated machine; physical HP Aruba 3810M SDN switches (Packard Enterprise, 2016b), which constructs a single and duplex bus topologies as shown in Figure 3.10; Four computers represent the communicated hosts; and, one-gigabit connections. Meanwhile, the emulated testbed consists of the same controller

mentioned above and Mininet emulator, which hosted on the single server. Mininet emulated single and duplex bus topologies as well with FastEthernet speed connection 1000Mbit, (see Figure 3.10).

All machines are running Windows 8.1, 64-bit with an Intel Core i7 CPU and 16 GB of RAM. Additionally, the server which hosted the controller and Mininet has an SSD hard drive.

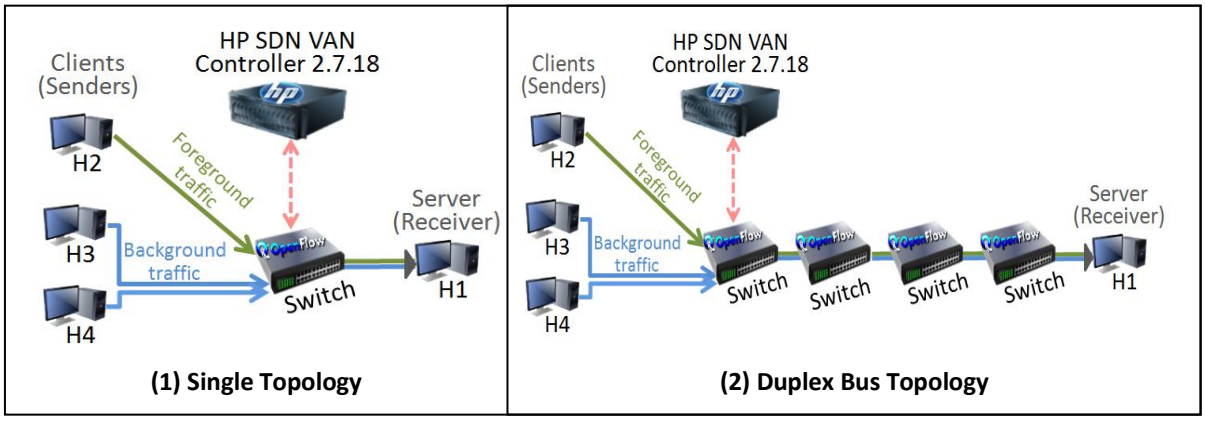


Figure 3.10: The used topologies in the tests of evaluating the performance metrics.

B) Measurement methods:

The Measurement method is divided into three types of measurement; they are "Latency", "Throughput" and "Jitter and packet losses" measurement. Every measurement type has several tests, which will be described below. For studying the impact of background traffic on foreground traffic, every test is performed three times. In the first time, the test measures the performance of SDN only with foreground traffic between H2 and H1, see Figure 3.10. In the second and third times, the foreground traffic was tested after saturating the network with background traffic, where both H3 and H4 sent background traffic to H1 with 250 and 375 Mbit. That saturated the network with total background traffic of 500 and 750 Mbit for the second and third times respectively. Next will be a brief demonstration of the three measurement types and their tests:

1) Latency measurement. Firstly, path latency is measured using: (a) the ping utility; (b) the proposed asynchronous latency measurement technique; and (c) (SYN-ACK/ACK) measurement technique, which is described in Section 3.3.2. The results of these three techniques are compared to show the accuracy of the proposed measurement technique. In ping and asynchronous latency measurement technique, the test is performed for three windows sizes of packets 1.5, 10 and 65 Kbytes. The three

windows sizes help to discriminate SDN behaviour for small, medium and large application packets. Whereas, for (SYN-ACK/ACK) measurement technique, the test is performed for only a small window size of packets because the window size always starts small through the three-handshake process of TCP connection. The results represent the average value of RTT for twenty pings or TCP probe packets.

Secondly, the connection flow-setup latency is found from the RTT of the first packet of each flow. It was extracted from ping and the proposed asynchronous technique is the same as the procedure mentioned above.

2) Throughput measurement. The throughput is tested for TCP and UDP traffic with four different windows size of the packets. This time, the test is done with 1.5, 10, 65 and 150 Kbytes as a window size for foreground traffic. The reason for adding a window size of 150 Kbytes is that the UDP traffic does not achieve the maximum throughput with a window size of 65Kbytes on HP Aruba switch, see Figure 3.13-b. The results averaged the throughput of a twenty TCP/UDP connection transmitted for 120 seconds using Iperf tool.

3) Jitter and packet losses measurement. Both of them were obtained for UDP traffic with four different windows size of the packets, similar to the throughput measurement. The results were calculated the average of twenty UDP connections lasted for 120 seconds using Iperf as well.

3.3.4. Results Evaluation

This section will discuss the results which were obtained from the practical implementation of the experimental traffic on single topology and duplex bus topology.

1) The Tests of the Single Topology:

The results of the single topology will present the four performance metrics ("path latency", "connection flow-setup latency", "throughput" and "jitter and jacket losses rate") to show the difference in the behaviour of the physical and emulated SDN equipments.

A) Path latency. The proposed asynchronous technique (Figure 3.11-b) show more accurate results than ping measurements (Figure 3.11-a). The proposed technique shows lower latency values which increase linearly with the increment of network background load, while ping shows higher and unstable latencies. The SYN-ACK/ACK

technique (Figure 3.11-c) shows comparable results to the results of the proposed technique but only for a small window size of packets. That verified the accuracy of the results of this work.

In the results of the proposed technique, HP Aruba switch occupied latency (red-line) were ten times larger than OVS latency (green-line) for the small window size of packets (1.5 Kbyets). Meanwhile, this difference reduces to four-times of latency in a window size of 10 Kbyets and becomes only two times the latency for a window size of 65 Kbyets.

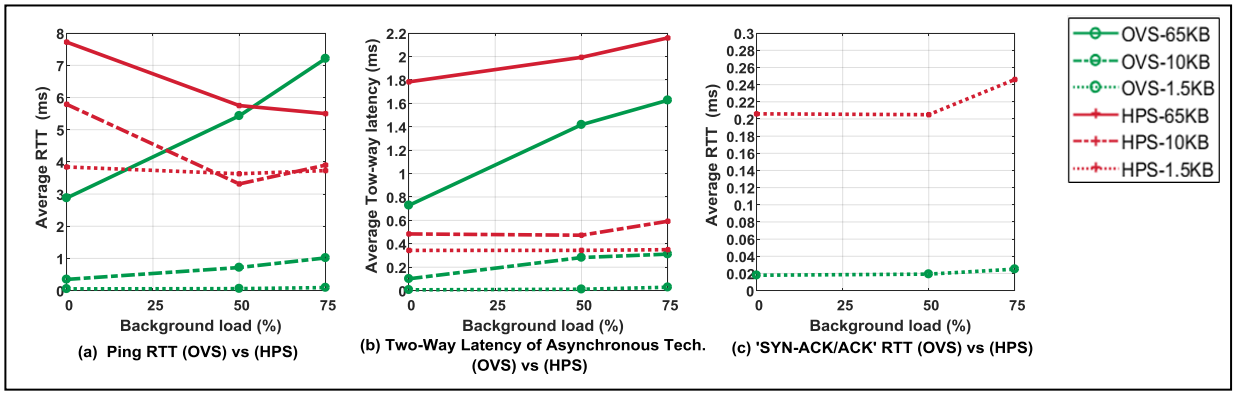


Figure 3.11: Two-Way Latency measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) in three different techniques.

B) Connection flow-setup latency. The connection flow-setup latency of the small window size of packets and zero background load was around 280ms for HP Aruba switch (red-line) and 3ms for OVS (green-line), when computed by the proposed asynchronous technique, see Figure 3.12-b. Diversely, connection flow-setup latency got around 400ms for HP Aruba switch (red-line) and 3.5ms for OVS (green-line), if extracted from ping RTT, see Figure 3.12-a. The difference in latency between the two techniques resulted from removing the processing latency of the second end from the flow-setup latency. That is proving the results validity of the asynchronous latency measurement technique specifically for the physical testbed.

Additionally, it is feasible from Figure 3.12 that, the difference in connection flow-setup latency between the real network and Mininet emulator is very big. The latency variation ranges from 100 to 40 times of flow-setup latency for small to the large window size of packets. Finally, all latencies increased with the rising of the background traffic of the network.

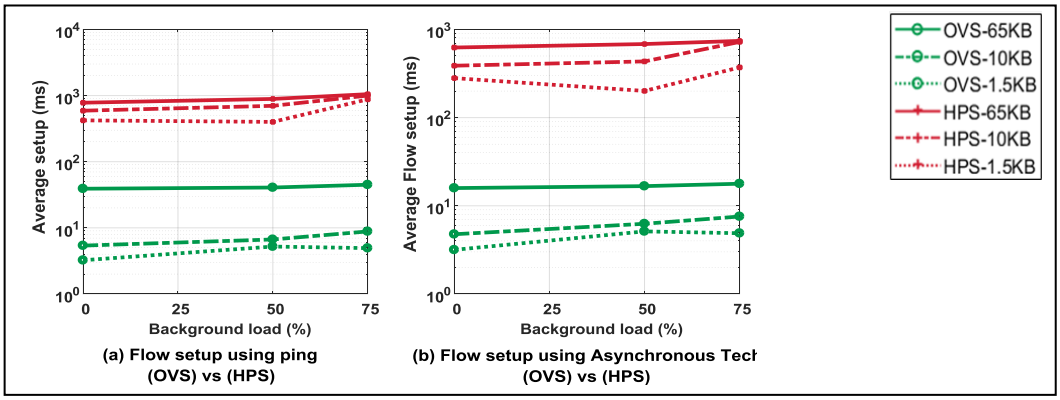


Figure 3.12: Flow setup Latency measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) in two different techniques.

C) Throughput. As displayed in Figure 3.13, the physical and emulated SDN provide different throughputs. However, the links of both testbeds were configured with the same bandwidth. On the one hand, HP Aruba generates very poor TCP throughput with a small window size of packets while OVS presents higher TCP throughput. On the other hand, HP Aruba provides the best TCP throughput with a large window size of packets.

The throughput of UDP traffic changed a lot in HP Aruba switch according to the window size of packets while a small variation of UDP throughput occurs in OVS. TCP/UDP throughput degraded when the background traffic of network is boosted.

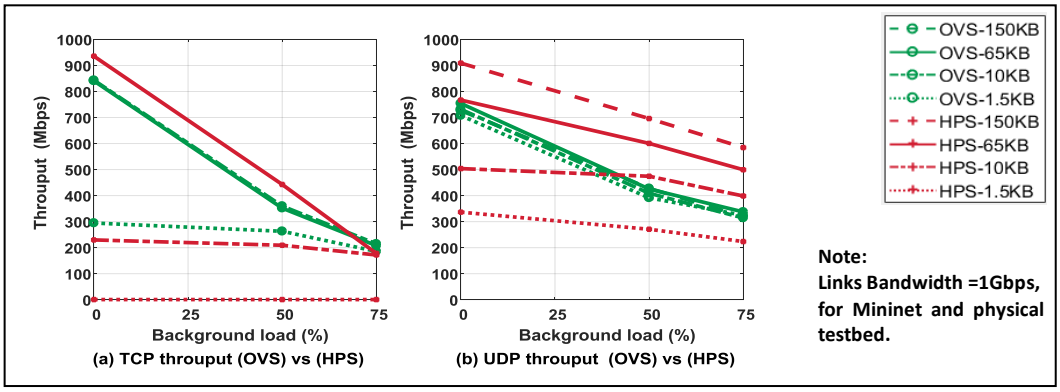


Figure 3.13: Throughput measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) for TCP and UDP traffic using Iperf.

D) Jitter and packet loss rate. The tests showed that OVS possess a lower jitter and packet loss rate than HP Aruba switch. However, the jitter and packet loss rate of HP Aruba are more responsive to the change of window size of packets and network background traffic, see Figure 3.14.

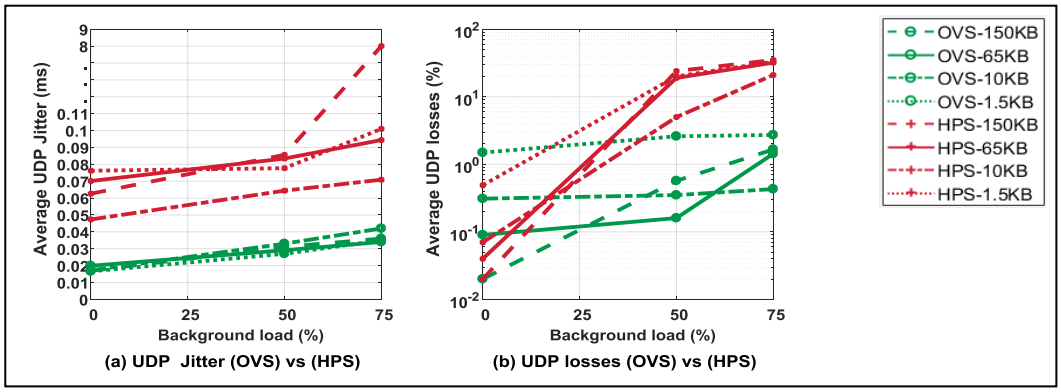


Figure 3.14: Jitter and loss packet rate measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) for UDP traffic using Iperf.

2) The Tests of the Duplex Bus Topology:

The results of the duplex bus topology will present only the two performance metrics ("path latency", and "connection flow-setup latency"). This is because the latency metrics increase with increasing the number of switches, while the other two metrics ("throughput" and "jitter and jacket loss rate") keep a similar behaviour for the single and duplex bus topology. Also, the latency metrics are the focus of this evaluation.

A) Path latency. Similar to the results of the single topology tests, the ping measurements have less precision in determining the communication latency (see Figure 3.15-a). While, the proposed asynchronous technique and SYN-ACK/ACK technique (see Figure 3.15- b and c), demonstrate better latencies in terms of the accuracy and the response to the changes of the communication conditions (packet size and background load).

Also, the difference in the behaviour of the latency between the physical and emulated switches is similar to the one in the tests of the single topology, but it only owned higher values.

In the results of the proposed technique, the latency of the HP Aruba switch (red-line) in these tests is higher than itself in the tests of the single topology. For example, the connection with a small packet size and zero background load has (0.97155 ms) in these tests, while it is only (0.3435 ms) in the single topology. This helps to calculate the forward latency at each switch (when the forwarding rule is already installed and it can directly forward the packet) from the difference between the results of the single and duplex bus tests, more details will be explained in Section 3.3.5.

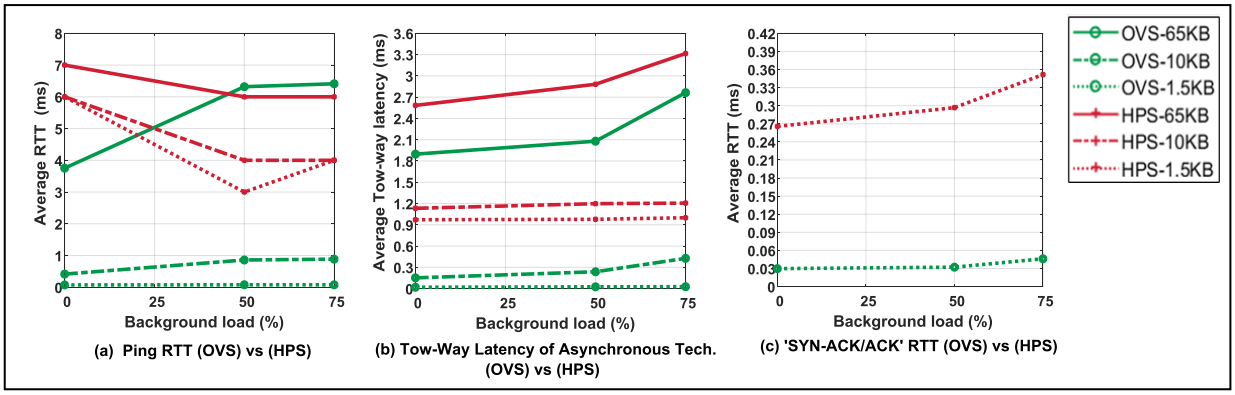


Figure 3.15: Two-Way Latency measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) in three different techniques.

B) Connection flow-setup latency. Figure 3.16 shows that the ping results are larger than the results of the asynchronies technique due to the extra latency of the destination side.

Also, Figure 3.16-b, demonstrates that the connection flow-setup latencies of the physical switches are larger than the latencies of the OVS by a big amount, which reaches to 100 times. It also, exposes that the latencies in these tests are larger than their peers in the tests of the single topology in proportional values.

Finally, the comparison of the connection flow-setup latency of these latencies with the latencies of the single topology testes indicates an identical attitude toward the packet size and background load. Also, it helps to compute the latency at each switch (when installing the forwarding rule for the first packet of the flow).

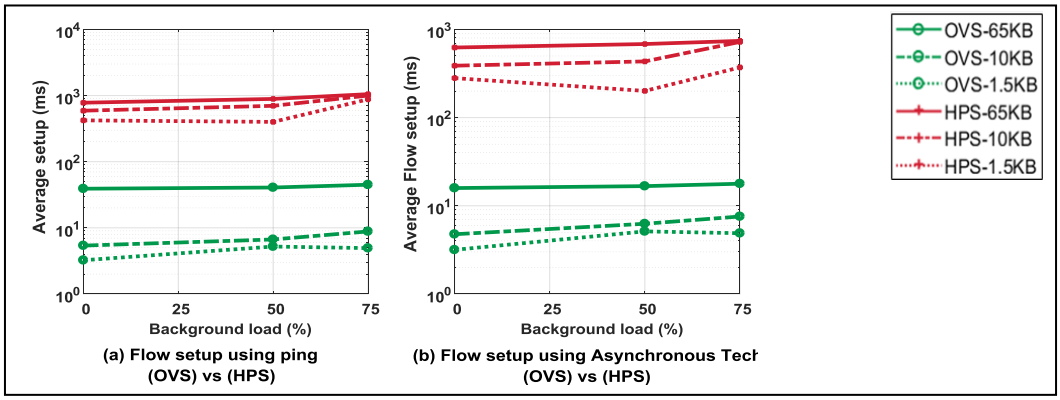


Figure 3.16: Flow setup Latency measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) in two different techniques.

3.3.5. Formulating a New Latency Metrics and Equation for SD-WAN

Examining and measuring the SD-WAN latency leads to defining new latency metrics and formulates the latency equation. This section explains: the motivation for creating the new metrics; Then, the steps of creating the connection; After that, the new metrics and their components; Flows by, formulating the latency equations; Finally, computing the value of every latency components from the testes results and equations.

1) The motivation of using the new latency metrics

As mentioned in Chapter one, Section "1.3. Research Methodology", the previous placement researchers only compared some latency features which were the focus of the optimisation process, before and after applying their placement. For example, the switch-controller latency, the inter-controller latency and the flow-setup latency (see chapter two Section 2.3.1). The flow-setup latency is defined as the summation of the latencies for sending the first packet of flow from the switch to the controller, processing it in the controller and installing its forwarding-rules back to the switch's flow-table (Jimenez *et al.*, 2014). All previous latencies represent only the individual delays at the switch or controller and do not demonstrate the latency of the complete path of the flow. Therefore, He *et al.* (2017) models the end-to-end flow-setup latency, which calculates the latency of transferring the first packet along the complete path from the sender to the receiver computers for single flow only. In He's line of thought, this research demonstrates that even the end-to-end flow-setup latency does not show the complete latency for the full process of sending a packet between any two hosts. That was the motivation to compute ***a new latency metric which is the 'connection flow-setup latency' and its features which are the 'connection control-path latency' and 'connection data-path latency'***. ***This introduces a clearer demonstration of the controller placement effect on the latency of the flow-setup time for connecting two communicators.*** The connection flow-setup latency is defined as the summation of all the time fractions for sending the first packet from the source-host to the destination-host, which is required to set the connection flow-rules of three flows in order to perform the sending operation. The connection control-path latency is restricted by the time summation of transferring the control packets that lead to complete the sending operation. Finally, the connection data-path latency includes the time of forwarding these packets through the path between the two hosts. Before defining the new metrics,

it is important to understand and define the steps of creating a connection, as explained in the next section.

2) Defining the connection steps

Sending a packet from the source (Src.) host to the destination (Dst.) host starts by requesting the Media Access Control (MAC) address of the destination computer (Plummer, 1982) (Goralski, 2009). Then, the original packet is constructed and sent. Therefore, the process of sending the first packet consumes three flows as demonstrated in the Figure 3.17 below. The first flow is the MAC request, the second flow is the MAC reply, and the third flow represents the first packet that is intended to be sent to the destination. However, the fourth flow is responsible for acknowledging the receiving of the first packet at the destination, which does not consider its time as part of the connection flow-setup latency (marked by shaded rectangular in Figure 3.17). Regardless of the type of the first packet, it is not important because the flow-rules are set according to its type. For example, if it is a TCP packet, then the TCP three handshake could be performed without the need for new flow-rules as its flows are already defined in the two directions.

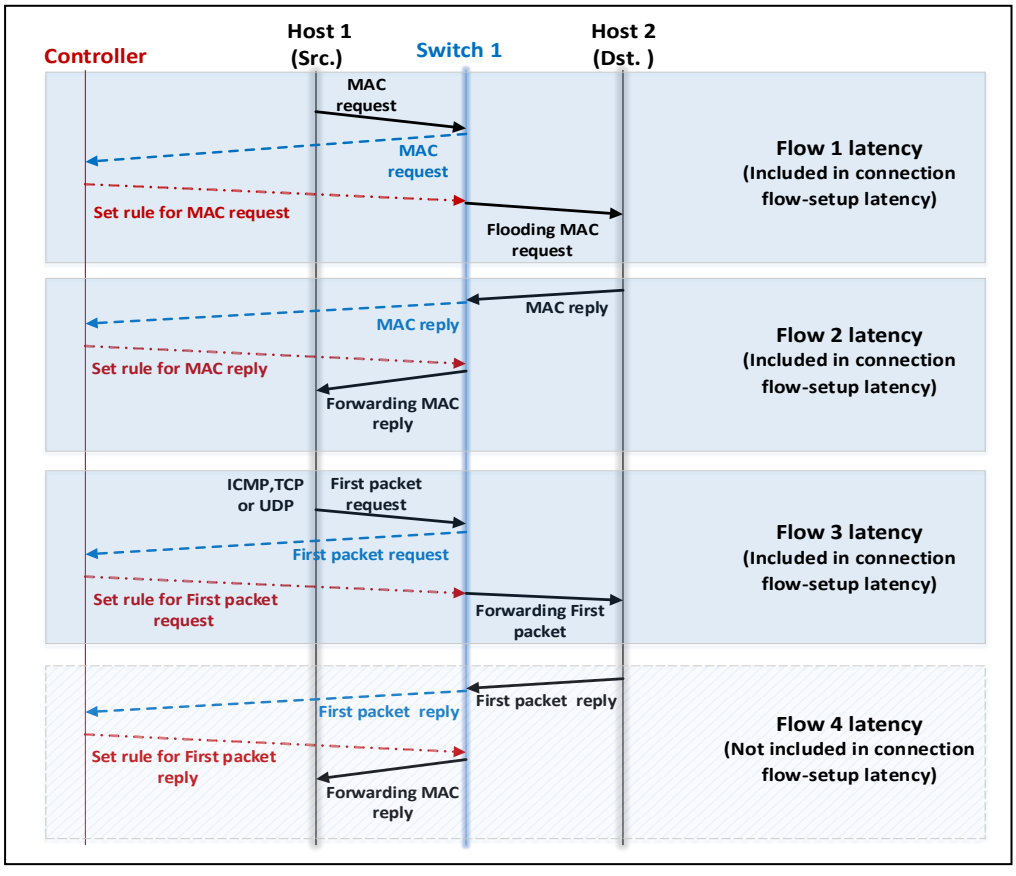


Figure 3.17: The steps of connection flow-setup latency.

3) Defining the latency components:

The process of sending the first flow starts when the packets arrive at the switch, which is connected to the source-host (see Figure 3.18). The switch then sends a flow-request to the controller through the control-path (red-dashed-line), which produces a delay that compounds of the Control-Path Switch (CPS) delays and the Control-Path Link (CPL) delays. After that, the controller computes the flow-rules in time called Controller-Delay (CD). Then, the controller uses the control-path, which connects itself to every switch to send the generated flow-rules simultaneously to all the switches on the data-path between the two hosts. Finally, the first packet of the flow is passed through the data-path (green-dashed-line), which creates a delay that is composed of the Data-Path Switch (DPS) delays and the Data-Path Link (DPL) delays.

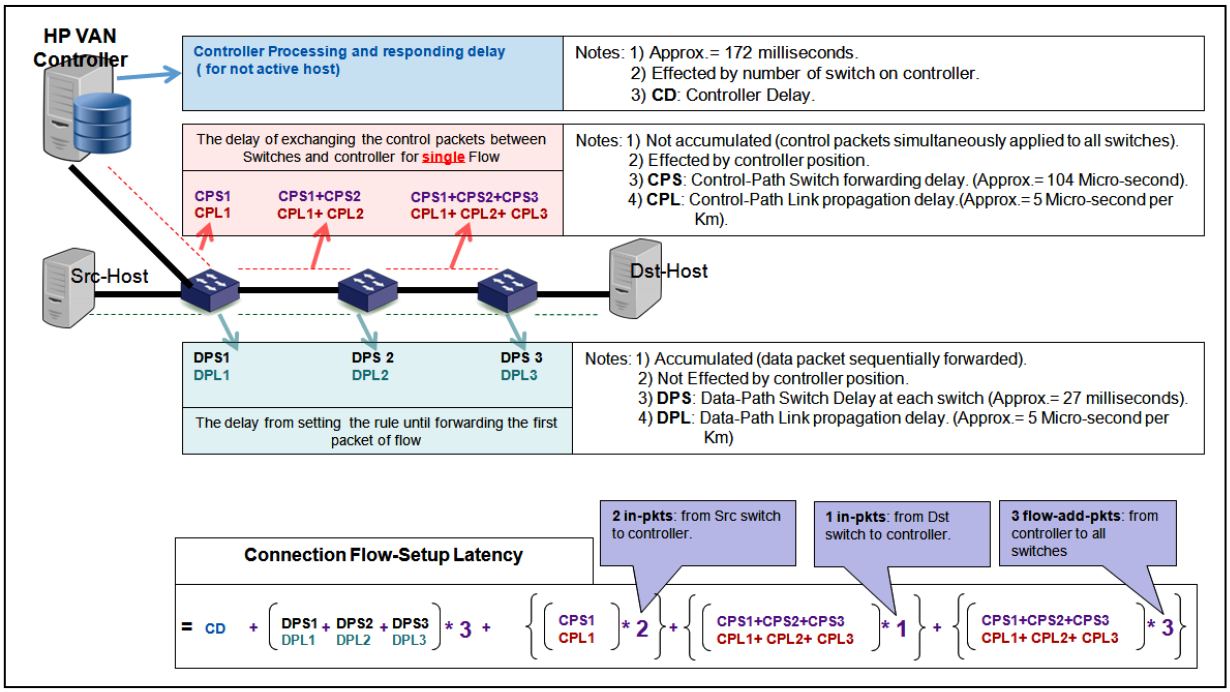


Figure 3.18: The outcome of examining the latency optimisation.

The complete delay of **passing one flow** equals the summation of **the controller-delay, the control-path delay** (the flow-request delay, flow-rules installation delay) and **data-path delay** (the delay of forwarding the first packet along data-path), as demonstrated below:

1. The controller delay is the complete delay at the controller itself.
2. The flow-request delay is the delay of the control-path from the source switch to the controller (**CPS1+CPL1**).

3. The flow-rules installation delay is the delay of the control-path from the controller to the farthest switch in the data-path of the flow (**CPS1+CPS2+CPS3 + CPL1+ CPL2+ CPL3**), because all flow-rules are send simultaneously.
4. The data-path delay is the accumulated delay of the switches and links of the data-path **DPS1+ DPS2+ DPS3 + DPL1+ DPL2+ DPL3).**

Finally, the equation in Figure 3.18, demonstrates the connection flow-setup latency, which includes the delays of the three flows (complete connection), but for a single controller. This equation will be explained in detail in the next section for a more complete view of SD-WAN, when it has multiple controllers.

4) Connection control-path latency

It is the summation of all latencies of control-packets, which are exchanged between the switches and controllers to install the required flow-rules for the three flows, which are: the switch latency, link latency and inter-controller latency. The control-path latency is represented by the red-dashed-line, which is located above the switches in Figure 3.19.

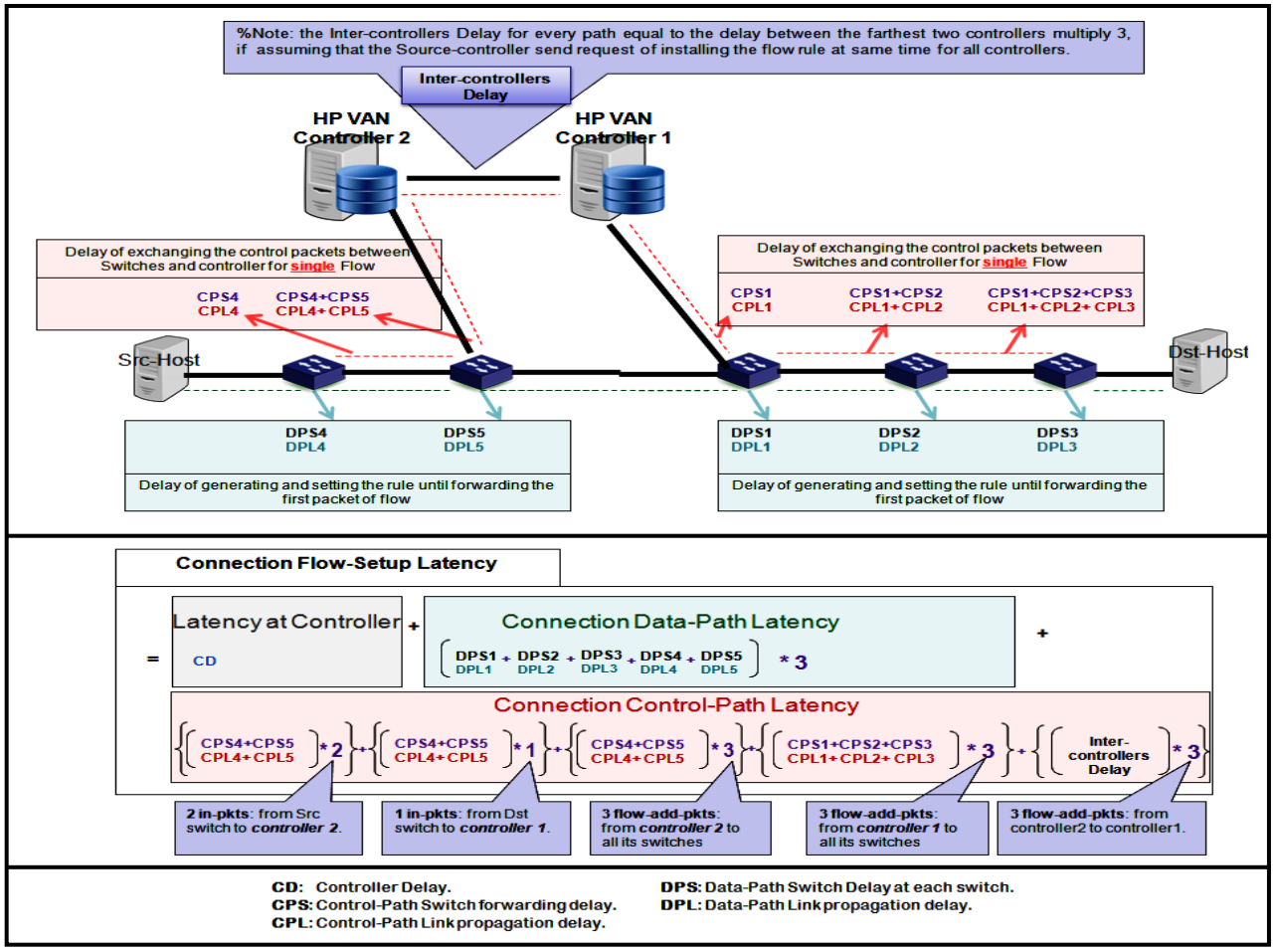


Figure 3.19: An example of SDN to demonstrate the connection control-path latency and connection flow-setup latency.

The connection control-path latency is also presented as a mathematical expression in the last five brackets (red-shaded-rectangular) of the equation of the connection flow-setup latency in Figure 3.19 above. The latencies of these five brackets are as follows:

1. First bracket: ***Two flow requests*** from the source switch to its controller (for sending MAC request and original packets in the forward direction)
2. Second bracket: O***ne flow request*** from destination switch to its controller (for sending MAC reply in the opposite direction);
3. Third and fourth brackets: Every controller requires installing ***three flow rules*** in data-path switches to pass the required packets in two directions.
4. Last bracket: The ***inter-controller*** latency. It is comprised of three packets, two of them passing the flow-rules from the first controller to another one (to forward MAC request and original flow), while the last packet carries the flow-rules from the second controller in the opposite direction (to forward the MAC reply).

5) Connection flow-setup latency

It consists of the controller delay (grey-shaded-rectangular), the connection data-path delay (green-shaded-rectangular) and the connection control-path delay (red-shaded-rectangular) as interpreted in Figure 3.19 and the above section.

6) Computing the latency components

The computation of the latencies from the results of the conducted tests are based on the following assumptions:

1. The delays of the three flows (Mac request, MAC reply and desired flow) are considered equals.
2. The propagation delays of the links of the physical testbed are ignored because they are only 5.5 ns/m for Ethernet (cat5) links (Beckhoff Automation GmbH & Co. KG, 2017), and the length of the used links was only one meter.
3. The propagation delays of the fibre optic links considered from 3.3 μs/Km for the new Photonic crystal fibres (PCFs) (Brosi, 2008)(Kawanishi *et al.*, 2012)(Kawanishi *et al.*, 2013), to 5 μs/Km for the standard optical fibres (Bobrovs *et al.*, 2014). Therefore, the CPL and DPL propagation delay equal the multiple of the fibre optic delay (5 μs/Km) by the length of the link.

After that, two types of computation are performed to compute the other latency components that occur, when setup the first connection and when forwarding the data over this connection after that.

The explanation of computing the latencies of forwarding data over established connections, will come first because some of its latency components are needed when computing the latencies of setup the connection.

A) Computing the latencies of forwarding data over established connections

The computation of this section finds the values of the following latency components: The latency of forwarding the control packets (OpenFlow packets) through the switches of the control path, which is CPS latency; Also, the outcome of this computation represents the latency of forwarding the data packets through the switches of the data-path, when the switches have the flow-rules (not the first packet of the flow), which are approximately equal to the CPS latency. The steps for performing this computation are as follows:

1. Obtaining the latency of forwarding the packets of small windows size (1.5 KB) for the single topology (343.5 µs) and for duplex bus topology (971.55 µs), from the tests of the asynchronies measurement technique on the physical testbed (see Figure 3.11-b and Figure 3.15-b). It is noteworthy that these latencies are the two way latencies.
2. Subtracting the latency of the single switch topology from the latency of the four switches of the duplex bus topology to obtain the residual latency of passing the packet through the last three switches. Then, dividing the results of the previous step by three to find the latency at one switch (for two ways) (see Equation 3.7 below). Subtracting these two latencies removes any additional latency which could be produced by the communicators' interfaces and keep only the delay of forwarding the packet at the switches.

$$Two\ ways\ latency\ of\ \ forwarding\ a\ packt\ at\ one\ switch = [latency\ of\ linear\ topology(4sw\ )] - [latency\ single\ topology(1sw\ )]\ /\ 3 \quad (3.7)$$

971.55- 343.5 = 628.05/3 =209.35 µs two ways latency of forwarding a packet at one switch

3. Dividing the two ways latency of one switch by two to get the one-way latency (see Equation 3.8 below).

$$CPS\ latency\ (Equals\ one\ way\ latency\ of\ one\ switch) = Two\ ways\ latency\ of\ one\ switch\ /2 \quad (3.8)$$

***CPS latency* = 209.35 / 2= 104.675 µs one-way latency of forwarding a packet at one switch**

The value of CPS latency demonstrates that the latency at a single node is larger than the propagation latency of 20Km according to the standard fibre optics specifications mentioned above.

B) Computing the latencies of setup the connection

This computation calculates the values of the following latency components: The latency at each switch on the data path when passing the first packet of the flow (DPS latency); Also, It computes the controller delay (CD). The steps for performing this computation are as follows:

1. Obtaining the latency of forwarding the first packet of small windows size (1.5 KB) for the single topology (280.448 ms) and for duplex bus topology (603.514 ms), from the tests of the asynchronies measurement technique on the physical testbed (see Figures 3.12-b and 3.16-b). It is noteworthy that these latencies are the ***two ways latencies of two flows*** (MAC flow and desired flow).
2. Subtracting the latency of forwarding the first packet of the single switch topology from the latency of the four switches of the duplex bus topology to obtain the residual latency of passing the first packets through the last three switches. Then, dividing the results of the previous step by three to find the latency at one switch (***for two ways of two flows***) (see Equation 3.9 below). Subtracting these two latencies removes any additional latency which could be produced by the controller and keep only the delay of forwarding the first packet at the switches.

$$Two\ ways\ latency\ of\ forwarding\ the\ first\ packets\ of\ two\ flows\ at\ one\ switch = [latency\ of\ linear\ topology(4sw)] - [latency\ single\ topology(1sw)] / 3 \quad (3.9)$$

603.514 - 280.448 /3 = 107.688 ms two ways latency of forwarding the first packets of two flows at one switch

3. Dividing the two ways latency at one switch by two to get the one-way latency (see Equation 3.10 below).

$$One\ way\ latency\ of\ forwarding\ the\ first\ packets\ of\ two\ flows\ at\ one\ switch = Two\ ways\ latency\ of\ forwarding\ the\ first\ packets\ of\ two\ flows\ at\ one\ switch/2 \quad (3.10)$$

107.688 /2 = 53.844 ms one ways latency of forwarding the first packets of two flows at one switch

4. Dividing the one ways latency of two flows at one switch by two to get the one-way latency of one flow (see Equation 3.11 below).

$$DPS\ Latency = One\ way\ latency\ of\ forwarding\ the\ first\ packets\ of\ one\ flow\ at\ one\ switch = One\ way\ latency\ of\ forwarding\ the\ first\ packets\ of\ two\ flows\ at\ one\ switch/2 \quad (3.11)$$

DPS latency= 53.844 /2 = 26.922 ms one ways latency of forwarding the first packet of one flow at one switch

5. Finally, the CD is calculated using the formula shown in Figure 3.20 and Equation 3.12.

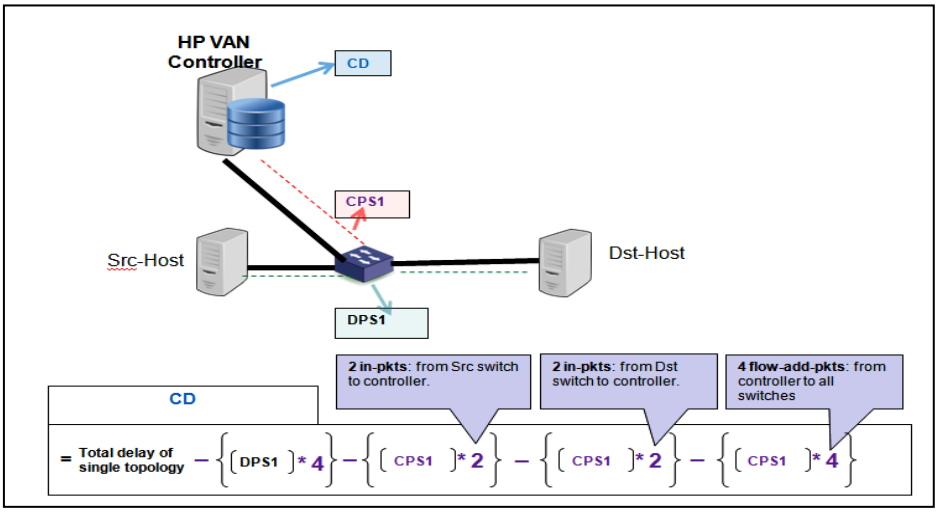


Figure 3.20: The formula of calculating the controller delay.

$$CD = [\text{latency of forwarding the first packet of single switch topology}] - \{DPS1 * 4\} - \{CPS1 * 8\} \quad (3.12)$$

CD= 280.448 - {107.688} - {8*104.675/1000} = 172.76 - 0.8374= 171.9226ms

All the above latency components are applied to compute the connection flow-setup latency using the formula which is presented in Figure 3.19.

3.4. Developed Technique to Measure the Latency of OpenFlow Packets Using the Controller

This section measures the latency of the OpenFlow packets that are exchanged between the controller and its switches. This latency is represented by the control path latency, which consists of all the CPSs and CPLs latencies that are resulted by the switches and links of this control path. These latencies are calculated from the results of the previous asynchronies technique, but by using the hosts in data-plan and normal data packets. Therefore, they are measured this time using the controller and OpenFlow packets to prove the validity of the previous assumption, which considered the DPS latency of forwarding the data packet (not first data-packet) and CPS latency of forwarding the OpenFlow packet through the switch are almost equal.

It is noteworthy that, to learn about the related work of the latency measurement technique, see Section 3.3.1. Furthermore, the testbed description could be found in Section 3.3.3. Also, it is possible to revise the latencies definition in Section 3.3.5.

3.4.1. Description of the OpenFlow Latency Measurement Technique

This technique using the controller machine to capture the time of sending and receiving the OpenFlow symmetric packets, which are continuously exchanged between the controller and its switches every five seconds. The computation is performed on the captured traffic of these symmetric packets, which are captured for about five minutes. This technique could be described in two processes, which are: (A) The capturing process; and (B) The analysis process.

A) The capturing process: The steps of this process as follows.

1. Connect the duplex bus topology of four hp SDN switches and HP VAN controller, as shown in Figure 3.21.
2. Start Mininet VM on the same computer, which hosted the controller.
3. Start Wireshark from Mininet (to read OpenFlow header).

   #sudo Wireshark &
4. Start Capturing the interface eth0.
5. Use the (of) filter to display all OpenFlow traffic.

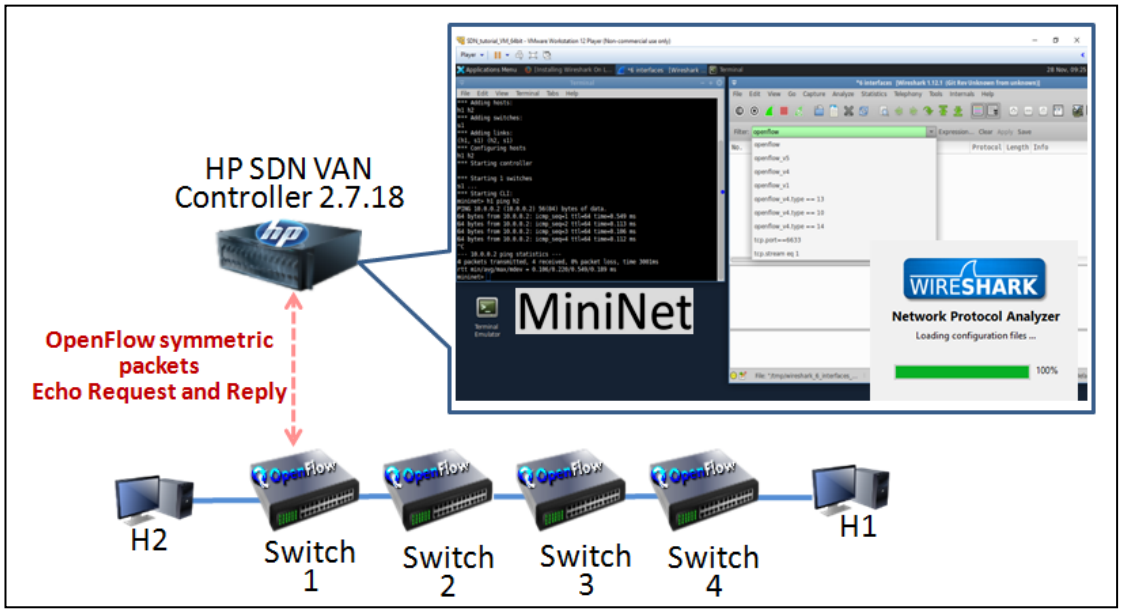


Figure 3.21: The testbed of implementing the OpenFlow latency measurement technique.

B) The analysis process: The steps to analyse the traffic and compute the latencies are as follows:

- The topology has four switches, so it is needed to analyze the traffic of each switch individually, to find the RTT between the controller and every switch.
- Each switch will reply to the controller when they receive a controller request. Therefore, to extract the RTT of bi-directional request-reply packets, two filters are required as following;

  1. Filtering the request from the controller to the switch (request C --> SW)
     (of) && (of10.echo_request.type == 2) && (ip.dst == 195.195.93.184)
  2. Filtering the reply from the switch to the controller (reply SW --> C)
     (of) && (of10.echo_reply.type == 3) && (ip.src == 195.195.93.184)

- After applying each filter, export the display packet as CSV (excel) file.

- Then, subtracting the Request time from the Reply time to calculate the RTT for each bi-directional request-reply between the controller and the switch (see Equation 3.13):

$$[RTT] = (reply\text{-}time) - (request\text{-}time) \quad (3.13)$$

- After that, subtract the RTT of the subsequent switch (sw_{i+1}), from the RTT of the previous switch (sw_i), to find the delta RTT of connecting the switch to its predecessor switch in the duplex bus topology (see Equation 3.14).

$$[Delta - RTT\ between\ sw_{i+1}\ and\ sw_i] = (RTTsw_{i+1}) - (RTTsw_i) \quad (3.14)$$

- Finally, dividing the Delta-RTT between switches by two to obtain the one-way forwarding latency from (SW2) to (SW1) (see Equation 3.15). This latency represent the CPS latency of (SW2), bearing in mind that, the CPL latency is ignored because it is very small (only 5.5 ns/m).

$$[CPS\ latency] = [Delta - RTT\ between\ sw_{i+1}\ and\ sw_i]/\ 2 \quad (3.15)$$

3.4.2. The Results of the OpenFlow Latency Measurement Technique

This section presents only the average value of RTT of every switch. Then, it demonstrates how to calculate the CPS latency from the resulting RTT of these switches, as below:

1. Switch 1:
   - $RTT_{average}$ = 0.000527912 second
2. Switch 2:
   - $RTT_{average}$ = 0.000683979 second
   - Apply Equation 3.14 to compute the Delta-RTT:

     Delta-RTT-sw2-1= 683-527=156 µs
   - Apply Equation 3.15 to compute the CPS latency:

     CPS latency sw2= 156/2= 156 µs
3. Switch 3:
   - $RTT_{average}$ = 0.000865087 second
   - Apply Equation 3.14 to compute the Delta-RTT:

     Delta-RTT-sw3-2= 865-683= 182 µs
   - Apply Equation 3.15 to compute the CPS latency:

     CPS latency sw3= 182 /2= 91µs
4. Switch 4:
   - $RTT_{average}$ = 0.00104125 second
   - Apply Equation 3.14 to compute the Delta-RTT:

     Delta-RTT-sw4-3= 1041 - 865 =176 µs

- Apply Equation 3.15 to compute the CPS latency:

CPS latency sw4= 176 /2= 88µs

Finally, the average value for the three DPS latencies is 85.6 µs. This value indicates that the difference between the results of this measurement and the previous measurement of the asynchronies technique (104.675 µs) is only 19 µs. This proves that the CPS latency of the data packet almost equals to the control packets. In addition, this small difference may occur because the synchronized packets between the controller and its switches are smaller than 1.5KB, and also they are continuing packets.

3.5. Summary and Conclusion

This chapter examines if the control packets affects the SD-WAN bandwidth. Also, it monitors and measures the latency metrics of creating the connection between two hosts, in order to define the complete latency formula and determine the values of latency metrics. This measurement identified a complete view of latency of SD-WAN and led to understanding to what extent the controller placement could affect this latency.

This chapter present examination as three contributions. The novel SFOP routing algorithm and its tests are explained. Then, the new asynchronies measurement technique is used to evaluate the performance of the physical and emulated SDN equipments. Consequently, another technique is employed to measure the latency of forwarding the OpenFlow packets on the switches of the control path.

In conclusion, the results of the SFOP routing algorithm demonstrate that the small control packets of OpenFlow do not degrade the network bandwidth, either with the shortest path or SFOP routing algorithms. Therefore, the research kept using the shortest path routing algorithm. It follows that, the outcome of the tests of the asynchronies technique algorithm contributes to the construction of the equation of the connection flow-setup latency and computes the value of its components from the practical results of the physical devices. Also, it helped to identify the required calibration of the performance metrics, which support the realism results of the emulated SD-WAN. Finally, the measurement technique for measuring the latency of the OpenFlow packets using the controller proves the validity of the CPS latency value, which is calculated by the previous asynchronies technique. Also, it proves the validity of the assumption that the forwarding latency is approximately equal for control and data packets.

CHAPTER FOUR
COVN PLACEMENT ALGORITHM

4.1. Introduction

This chapter presents the design of the novel COVN placement algorithm for SD-WAN with multiple virtual networks. It starts by restating the research problem that is identified by the thesis (missing a controller placement algorithm for multiple virtual networks) and presenting the main contribution (COVN placement algorithm). After that, Section 4.3 introduces the objectives, requirements and purpose of the new model. Then, Section 4.4 explains the possible methods to place the controllers of SD-WAN with multiple virtual networks in three different scenarios to validate the most appropriate solution, which is considered in the present thesis. Subsequently, Section 4.5 describes the usage and dynamic implementation of the COVN placement algorithm. The complete structure of the algorithm is then shown in Section 4.6. The design of the algorithm is demonstrated after that in two parts. Finally, the chapter ends with a summary and conclusion of the presented design.

4.2. Problem Statement and Solution

This section summarises the considered research of the controller placement algorithms and states the problem, with the proposed solution. As mentioned earlier in the literature review, the controller placement problem was developed to optimise several criteria of SD-WAN such as: minimising flow-setup latency, maximising the reliability, reducing the cost and power consumption and considering the multi-objective approach (see Chapter 2, Section 2.3.5). The present thesis investigates all the above models and ***finds the following problem,*** to the best of the author knowledge, all previous research of controller placement does not observe the inevitability of deploying the virtual networks over SD-WAN to extend its ability and increase its flexibility. ***Consequently, there is no single placement algorithm to dynamically place the controllers of SD-WAN with multiple virtual networks. Therefore, this thesis aims to design a novel controller placement algorithm, called COVN placement algorithm, to dynamically place the controllers of SD-WAN with multiple virtual networks to optimise their reliability, their flow-setup latency, their load balancing and their resources utilisation.***

4.3. Introducing the Objectives and the Requirements of the COVN Placement Algorithm

This section explains the objectives and requirements of the sliced SD-WAN. The last sub-section highlights the differences between the controller placement algorithm for SD-WAN and the one for SD-WAN with multiple virtual networks.

4.3.1. The Objectives of COVN Placement Algorithm

The COVN algorithm performs the following objectives: First, the flow-setup latency is minimised through reducing the latency resources such as: the link delay (distance between nodes), the number of hops between switch and controller, the inter-controller communication latency and the controller response time (restricting the capacity of the controllers); Second, the network resilience is optimised via including the rank of the nodes which host the controllers (number of joint paths from this node to all switches) and preparing the backup controllers; Thirdly, the load balance is provided by partitioning the SD-WAN into balanced clusters and assigning each cluster to the most appropriate controller by considering the distance and the current load of the controllers; Finally, utilizing the SD-WAN resources is achieved through assigning the minimum number of controllers for each VN**,** which reduces the number of active controllers and the used paths (switches-controllers and inter-controllers paths).

4.3.2. The Requirements of COVN Placement Algorithm

In order to set up the requirements of COVN placement algorithm, the sliced model of SD-WAN should be described, and some terminology which is used in this thesis should be defined. Running multiple virtual networks over a physical SD-WAN will slice it to create a more flexible model of the network which functions as several dedicated networks. *Hence, this requires placing the controllers of every slice independently.*

Each VN could have a different administration and serve a different sector of the community such as Education, Health and Telecom. One essential purpose of having the dedicated virtual networks is for utilising the usage of the physical resources by modifying the virtual topology according to the demands of the services (Ordonez-Lucena *et al.*, 2017). *Therefore, each VN can instantly make substantial changes in its topology due to the high flexibility of the software-based topology. Consequently, every VN needs to adjust its controller placement frequently, which requires a minimal computation complexity of a placement algorithm. The virtual controllers need to be adjusted dynamically in reaction to the changes of data-plane as will demonstrate in Table 4.2.*

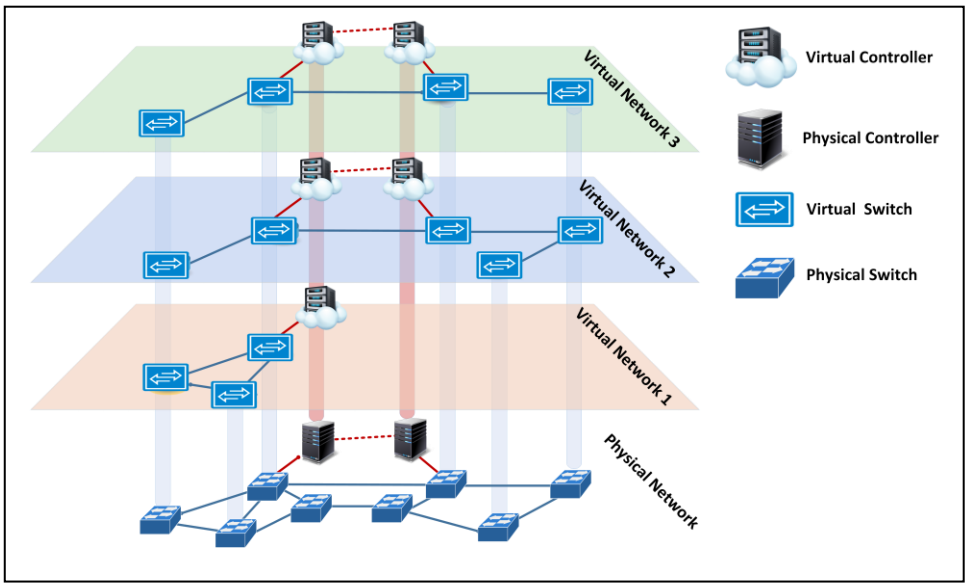


Figure 4.1: The SD-WAN with multiple virtual networks.

In this study, the researcher assumes that the control layer of SD-WAN is composed of logically centralised but physically distributed controllers (see Figure 4.1). The controllers are connected and collaborate. However, they are placed in various locations to avoid a single point of failure and to reduce the switch-controller communication latency. Usually, in SD-WAN the controller comprises of a server, with controller software running over it. Differently, in the sliced SD-WAN, the controller requires a powerful server which hosts multiple dedicated instances of controllers' software each serving one VN or a single controller software serving all virtual networks. In this thesis, the active server which is hosting the controller software is ***called the physical controller*** and the controller software that runs on the server (either the dedicated controller or a portion of the controller) is ***called the virtual controller*** as shown in Figure 4.1. *Possessing two types of controller requires a separation of the process of locating the* ***physical controllers*** *from the process of placing the* ***virtual controllers.***

Additionally, to study the usability of the existing placement algorithm with the sliced SD-WAN, this chapter presents the current placement algorithms according to their used approach into two groups, see the left side of Figure 4.2. The first group defines the controller node as a location which optimises the targeted metrics. This group searches the subset of nodes which could get the optimal value of the desired metrics, for example (Heller *et al.*, 2012). The second group splits the topology into clusters, then finds the optimal location for the controller of each cluster to optimise specific metrics for

instance (Zhang *et al.*, 2011). *Both groups end up with the controller's locations relating to a subset of nodes named as control domains or clusters, see the left side of Figure 4.2.*

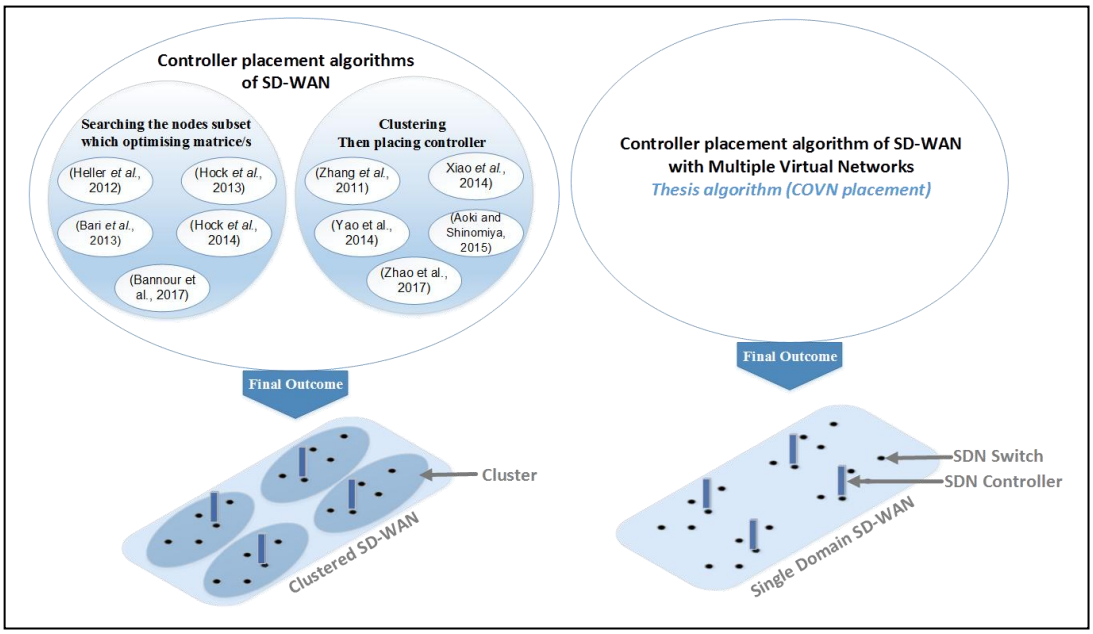


Figure 4. 2: The difference in the outcome of the existing placement and the COVN placement algorithms.

Placing the physical controllers according to the clusters of the physical topology does not provide an optimal controller placement because every slice has a different virtual topology and necessitates a different placement. *Therefore, the COVN algorithm requires the provision of a physical controller placement which is not related to the clusters of the physical topology to enable an efficient placement for each VN, see the right side of Figure 4.2*. The explanation of this assumption will be demonstrated in Section "4.4 COVN Placement Approach ".

From the above, it is possible to reorganise and summarise the requirements of COVN placement algorithm as follows:

1) COVN placement needs to place the physical controllers (active servers) by considering the prerequisites of the entire network (all virtual networks), such as the maximum load and the latency.

2) After that, the algorithm should distribute the virtual controllers (software controllers) over the physical controllers for every VN individually.

3) The frequent changes in virtual topology cause the virtual controller placement to be repeatedly applied, which requires the placement algorithm to have a low computational complexity.

4) The COVN placement algorithm has to solve a new problem, which is, the placement of the physical controllers which should not be dependent on the clusters' shape of the physical topology. This is because each VN could have a different clusters shape of its topology, the latter requires other locations of physical controllers in order to set the virtual controllers, see Section 4.4 for more details.

Finally, Table 4.1 displays the disparity in the requirements between the current and the CONV placement algorithms.

No.	SD-WAN placement algorithms	SD-WAN with Multiple Virtual Networks (CONV placement algorithm)
1	**Single network.**	**Multi-network.**
2	**Single step,** both the server and controller software placed together.	**Two steps**, placing server (physical controller), then placing the software (virtual controller).
3	**Less frequent placement**.	**More frequent placement** (lower computational complexity)
4	**Considering the shape of clusters of the physical topology to place the controllers.**	**Should not consider the shape of clusters of the physical topology to place the physical controllers.**

Table 4.1: The differences between the requirements of the placement algorithms of SD-WAN and CONV placement algorithm.

4.4. COVN Placement Approach

(The COVN placement approach of using the whole SDN network as a single domain)

It is notable that the existing placement algorithms determine the position of every controller in relation to a subset of nodes (cluster) in the SD-WAN topology. Whereas, the COVN approach needs only to identify the most optimum nodes (controllers' positions) in the graph without clustering the topology (see the method details in Section 4.6). This is regarding of coexisting of multiple virtual topologies that have different shapes, as it is mentioned as in Section "4.3.2. Define The Requirements Of COVN Placement Algorithm".

In order to solve the above problem, the COVN algorithm should place the physical controllers without portioning the network so that the VN can only use the required number of controllers regardless of the domains restrictions. Therefore, the COVN algorithm depends on using the whole SDN network as a single domain. It works to find the specific nodes, which have the best communication abilities by exploiting the graph theory, the details will be presented in Section 4.7. That means the selected nodes have less latency to communicate with all the other nodes of the network, less latency to

communicate with each other and more routes to connect to other nodes. The number of the specified nodes should be equal to the number of required controllers. In this way, it is possible to assign any number and combination of the N nodes (controllers) to control any network switches. The algorithm would not need to worry about assigning a specific switch to a specific controller because all controllers have communication abilities which are approximately similar to all other switches. That helps to assign only the required number of controllers to each VN. Also, it supports the dynamic change of the controller's number and location according to the needs of virtual networks.

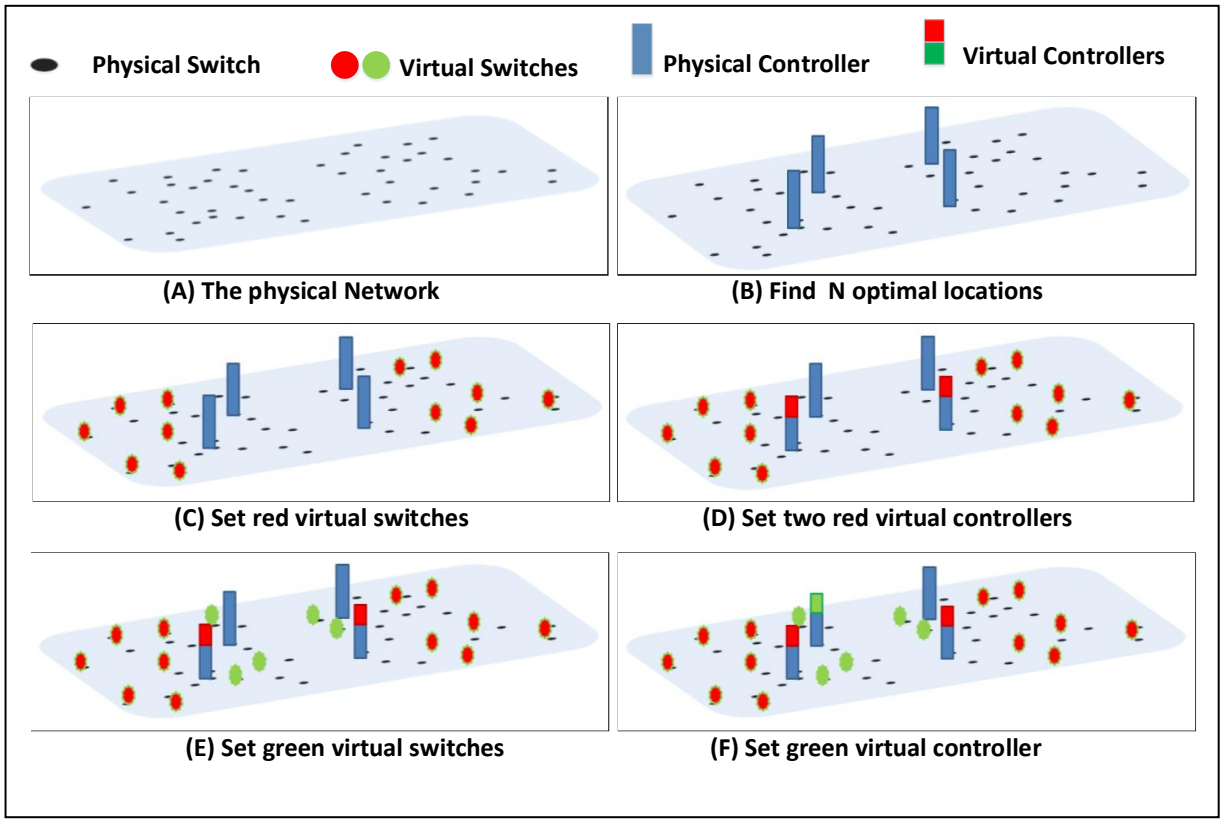


Figure 4.3: Placing the physical controllers using the COVN placement algorithm to optimise controller placement of virtual networks.

Figure 4.3: A and B present how to find the four optimal locations of controllers for all nodes of the network using COVN placement algorithm. In Figure 4.3: C and D, only two virtual controllers were assigned to manage the red VN. Since, the two controllers have enough capability to manage the whole red VN. This will reduce the number of virtual controllers to be used and reduce the inter-controller communication time and information. Similarly, for the green VN, one controller was sufficient to manage it (see Figure 4.3: E and F). All of this is now possible due to the new approach of the COVN placement algorithm. The resulting placement proves the validity of using this approach in this work.

4.5. The Usage and the Dynamic Implementation of COVN Placement

This section starts by summarising where and when COVN algorithm could be applied, then it shows why it is possible to apply this algorithm dynamically.

4.5.1. The Usage of the COVN Placement Algorithm

The controller's architecture of SD-WAN is either hierarchy architecture where all controllers are managed by the master controller or flat where all controllers have a similar role (Hock et al., 2013). In the hierarchy controllers, the placement algorithm is always executed by the master controller. While in flat architecture, an undetermined controller takes the master role to be responsible for performing the dynamic placement of controllers when needed. Afterwards, this controller may stay active or may be deactivated and transfer this role to the active one. Consequentially, the algorithm is always implemented in the master controller.

The second issue which needs to be addressed, is when should the COVN placement algorithm be applied. Applying this algorithm is dependent on the changes of SD-WAN status, as summarised in Table 4.2.

No.	Type of change	Description	Where	Action
1	**Big** Change	Increase/decrease the load of all network. → **More /less** than the processing capacity of physical controller/s.	Overall network.	Recalculate no. of Physical controllers. Replacement the Physical controllers.
		increase/decrease the number of virtual switches. →**More/Less** than the nodes capacity of physical controller/s.	Overall network.	Recalculate no. of Physical controllers. Replacement the Physical controllers.
2	**Small** Change	Increase/decrease the load of one virtual network load → **More /less** than the processing capacity of one virtual controller/s.	Over virtual network only.	Recalculate no. of virtual controllers. Replacement the virtual controllers.
		Increase/decrease the number of virtual switches →**More/Less** than the node capacity of virtual controller/s.	Over virtual network only.	Recalculate no. of virtual controllers. Replacement the virtual controllers.

Table 4.2: The summary of all the status of applying the COVN placement algorithm.

This Table demonstrates that there are two types of status changes, the big and small changes. The big change means the variation of load or number of virtual switches exceedes or less than the abilities of the current physical controller/s. The big change

leads to the implementation of the COVN placement algorithm overall the physical network and replaces the physical controllers. The small change defines the change of load or number of virtual switches within a single VN only, where the variation becomes over or under the capacity (load processing or a number of nodes) of the virtual controllers. This causes the COVN placement algorithm to be partially applied over this virtual network only.

Also, there are two different uses of the placement algorithm. Firstly, it could be used to optimise the performance of the network when the load or the number of virtual switches exceeds the ability of the controllers. Consequently, the placement algorithm is employed to increase the number of controllers to satisfy the new requirements of the network. Secondly, it may be used to optimise the utilisation of resources. Therefore, if the load or switch reduction equal to or more than the capacity of one controller, the placement algorithm is applied to reduce the number of controllers. Keeping the load or number of switches to less than the current controller's abilities could be efficient in term of performance of the network. However, it has a negative effect on the utilisation of resources. Therefore, the final decision about how to use the placement algorithm will depend on the network administrator.

4.5.2. The Dynamic Implementation of the COVN Placement Algorithm

The COVN placement algorithm is applied in two phases which are, computation and implementation. Computing the controllers' placement does not disturb the work of SD-WAN because the placement process is only applied after completing the algorithm computation. Therefore, the COVN placement algorithm does not have a critical time. However, the faster the recalculation benefits quicker optimisation of network behaviour*, which means it is acceptable to produce the decision in minutes but not in hours even for large SD-WAN*.

The second phase is implementing the placement decision (reassigning the switches from the current controller to another one). Even this process does not completely stop the work of SD-WAN for two reasons. One reason is that, the switches can keep forwarding the active flows based on the rules instilled before the reassigning process until all the packets of these flows are delivered. Another reason is, the dynamic provisioning of controllers by periodically assigning the switch to controller is developed based on OpenFlow 1.3 (Pfaff *et al.*, 2012), to be live (not noticeable) and safe

process in many works such as (Dixit *et al.*, 2013) (Bari *et al.*, 2013) (Auroux *et al.*, 2014) (Tuncer *et al.*, 2015a) (Yanyu Chen *et al.*, 2015) (Tao Wang *et al.*, 2016) (Tao Wang *et al.*, 2017) and (Cello *et al.*, 2017).

4.6. The Full Structure of COVN Placement Algorithm

The structure of the COVN placement algorithm is demonstrated in this section. The algorithm consists of: the initialising; the part-one (physical controller placement); the part-two (virtual controller placement); and the restarting of COVN algorithm, as shown below in Figure 4.4. The algorithm begins by reading the nodes of physical topology to identify their connectivity and the length of the links between these nodes and also the nodes positions. Also, it identifies the maximum capacity of load, which is the number of switches or the flow requests per second. The last task for the initialisation process is identifying the preferred network policy, for example, using the placement algorithm to optimise only the performance or both network performance and its utilisation of resources.

After that, the part-one of the COVN algorithm is performed, which is the placement of physical controllers. It calculates the maximum number of physical controllers needed (N) according to the maximum number of switches or their load. After that, it finds the optimal (N) locations to place the physical controllers according to the closeness and connectivity of these loacations.

Coming next, the part-two of the COVN algorithm which is the independent placement of virtual controllers of each VN. It places the controllers according to the number of virtual switches, their locations and the most suitable physical controller in term of communication latency, reliability against path failures and load processing capacity (to optimise the load balance among the physical controllers). Also, it shows how to manage the independent dynamic placement of each VN. If the number of virtual switches or its load is changed, the replacement algorithm of the virtual network is executed.

Finally, the reapplication conditions are performed in the restarting step. It keeps checking the controller status and comparing it to the identified parameters. Consequently, it reapplies the placement algorithm if the check has passive results to optimise the SD-WAN dynamically, see Figure 4.4 below.

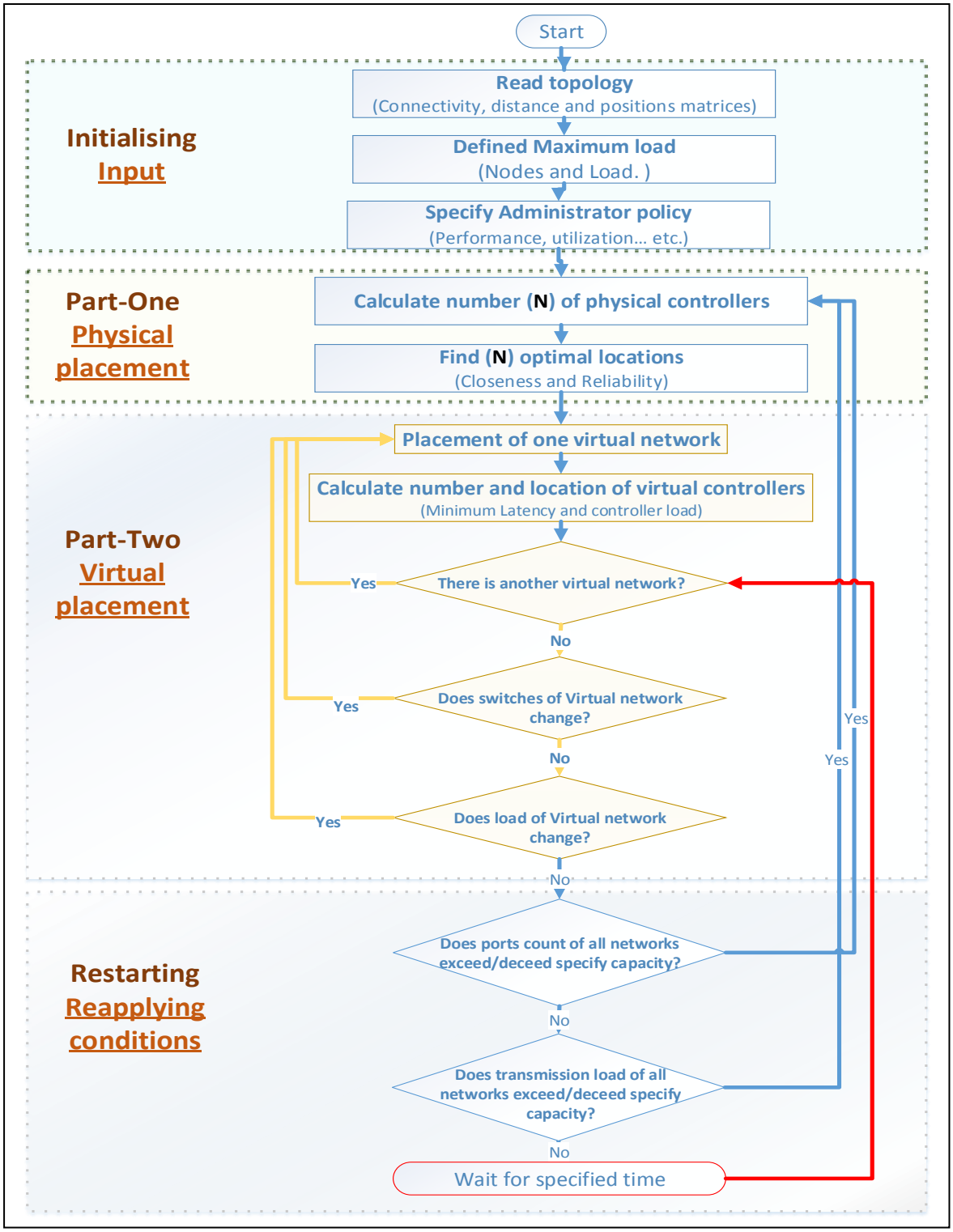


Figure 4.4: The structure of the COVN placement algorithm.

4.7. The COVN Placement Algorithm Part-One

The part-one of COVN algorithm is placing the physical controllers. This part calculates the number of active and backup controllers needed regarding the maximum load of the complete network. Then it identifies the controllers' positions by selecting the optimum nodes in the entire graph according to their closeness (latency) and connectivity

(reliability). The closeness indicates the nodes which possess the lowest latency to communicate with all other nodes of the network. While, the connectivity specifies the nodes which have the highest reliability against path failures with all other graph nodes. The following subsection demonstrates why combining the multi-objectives is used. After that, the sub-features of the latency and reliability is defined. Then, the design of part-one is explained. Finally, the computational complexity of this part is presented.

4.7.1. Trading-off versus Combining the Multi-Objectives (Latency and Reliability)

The objectives of placing the physical controllers are: optimising the SD-WAN flow-setup latency and its reliability against any failure (path and controller failures). However, process two of the methodology reveals that optimising the latency by improving the controller placement produces a small latency optimisation (see Section 1.3-B). The latency improvement is still considerable because every network has a different topology and purpose. For example, in the network which has sparse connectivity, reliability could be more important, while in a network with dense connectivity the flow-setup latency may act as the optimal target.

On the one hand, trading-off between them could offer an acceptable solution. On the other hand, combining these objectives with variable weight may introduce better placement. ***Therefore, this study tested both ways and found that combining the latency and reliability according to the method of the COVN algorithm provides a better placement than trading-off between them.***

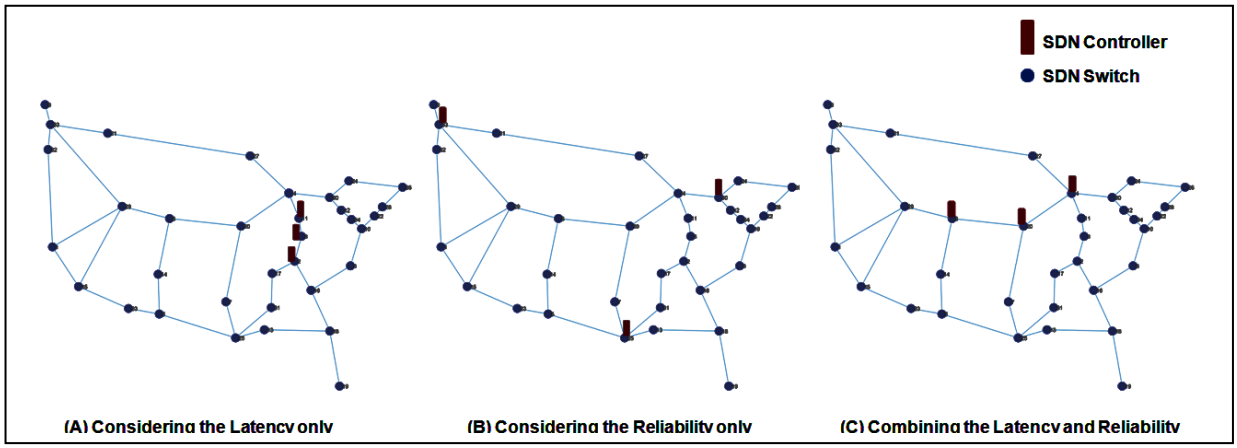


Figure 4.5: The controller placement according to the reliability alone, latency alone and Combining both of them.

In more detail, implementing COVN placement base on the latency causing all controllers to be set in the centre of the network (due to closeness centrality), which could provide

a good latency but could choose some unreliable nodes, see Figure 4.5-A. While, placing them with COVN algorithm according to the reliability allocates the controllers in the rubout nodes against path failure but could scatter them across the network which produces high flow-setup latency, see Figure 4.5-B. Therefore, combining both placements with specified weight leads to the selection of the reliable nodes which are located in the centre of the network, to get the optimal placement for this model, see Figure 4.5-C. The weight of latency to reliability placement differs according to the type of topology (see results in Chapter 6).

4.7.2. The Features of Latency and Reliability

The flow-setup latency has many features which could be improved in order to get the optimal flow-setup time. In the present research, the optimised features of the flow-setup time mainly consist of switch-controller latency, controller response latency and inter-controllers latency. All of them are included in the flow-setup latency minimisation as follows. The inter-controller latency is reasonably minimised without harmfully mounting the controller-switch latency. While optimising the controller response latency is kept to be improved in part-two of COVN placement. The interest in the inter-controller latency is based on its effective role in the flow-setup latency which is mentioned in several placement researches (see Chapter 2, Table 2.6 the points 3,4,8 and 9) and the direct observation from the practical results (see Chapter 6).

The reliability of controller and path failures are taken into account in this thesis. Controller failure is covered by preparing the backup controllers when calculating the controller's number. While, path failure is avoided by selecting the graph nodes which have optimal connectivity to the other nodes.

4.7.3. The Design of COVN Placement Algorithm- Part 1

This section describes the definitions, formulation and the complete design of part-one of COVN algorithm, which is the ***placement of the physical controllers*** (see Figure 4.4).

A) Definitions

- Network:

  The COVN placement algorithm considers the SD-WAN as an undirected graph ***G (N, L)***, Where (***N***) is the nodes of the graph and (***L***) is the links that are connecting them. The node count is $|N| = n$ and the length of links is $|L| = l$. The path (***P***) is

the sequence of nodes from Ingress node ($\boldsymbol{In}$) to Egress node ($\boldsymbol{En}$), $\boldsymbol{P} = (\boldsymbol{In} = \boldsymbol{n_i}; \boldsymbol{n_{i+1}}, \dots, \boldsymbol{n_k} = \boldsymbol{En})$, where $\boldsymbol{K}$ is the length of the path (its hop count) and $(\boldsymbol{n_i}; \boldsymbol{n_{i+1}}) \in \boldsymbol{N} : \forall \boldsymbol{i} = (\boldsymbol{1}, \dots, \boldsymbol{k-1})$. The length of link ($\boldsymbol{l}$) from node ($\boldsymbol{m}$) to node ($\boldsymbol{n}$) denoted by $\boldsymbol{l}(\boldsymbol{n}, \boldsymbol{m})$, where $\boldsymbol{n}, \boldsymbol{m} \in \boldsymbol{N}$ with associated delay ($\boldsymbol{\delta_l \geq 0}$). Finally, the controllers of SD-WAN symbols as ($\boldsymbol{C}$).

- Input:
  - The maximum capacity of the physical controller as the number of switches ($\boldsymbol{\sigma^C}$).
  - The maximum capacity of the physical controller as requests per second ($\boldsymbol{\rho^C}$).
  - The number of physical switches ($\boldsymbol{n}$).
  - The Load matrix ($\boldsymbol{Ld^N}$), which has the load of every switch.
  - The connectivity matrix used here is the adjacency matrix ($\boldsymbol{AdjD}$).
  - The weight of the node latency to communicate all other nodes ($\boldsymbol{\omega_L}$) against the weight of the node reliability in network ($\boldsymbol{\omega_R}$).

B) The Formulation of finding the Number of Controllers

Calculating the number of the physical controllers ($\boldsymbol{C}$) according to controllers' capacity ($\boldsymbol{\sigma^C}$ or $\boldsymbol{\rho^C}$) and the maximum load, which are either the number of switches (see Equation 4.1) or the requests per second (see Equation 4.2), as following:

$$C = N/\sigma^C \qquad (4.1)$$

$$C = \left(\sum_{i=0}^{N} Ld_i^N\right) \Big/ \rho^C \qquad (4.2)$$

C) The Formulation of finding the controllers Locations

The targeted controllers' node should have the minimal average latency (hop count) and the maximum average reliability (joint paths) to all the other nodes of the graph. The COVN placement algorithm is doing that using the graph theorem to count the paths between the vertices (Rosen, 2012) on page 688. This theorem considers the adjacency matrix ($\boldsymbol{A}$) as a map of a graph that indicates the number of the paths with one link of length between every pair of nodes. Consequently, powering this adjacency matrix to the second rank ($\boldsymbol{A^2}$) finds the number of paths which have their length equal to two links between every node pair, and repeating the same process to compute the number of paths of three links in length, as shown in Figure 4.6.

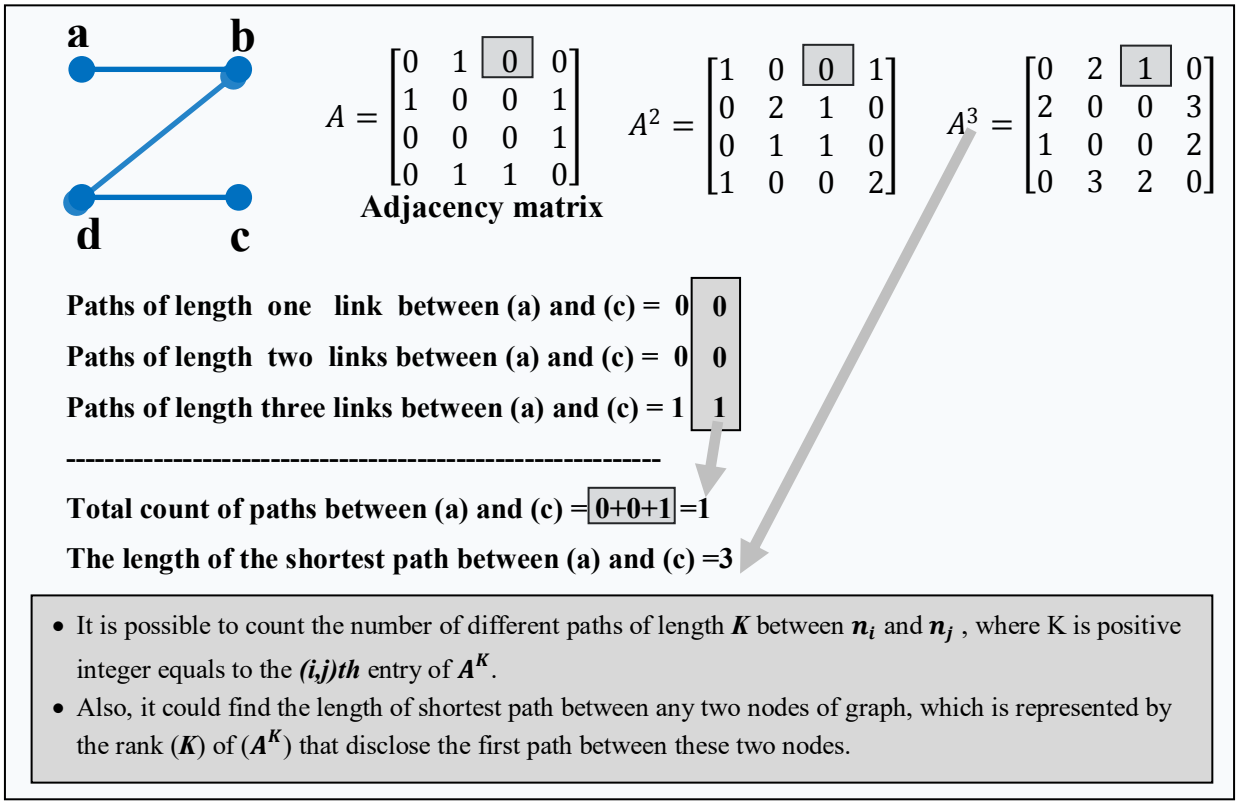


Figure 4.6: The graph theorem to find the count of paths between two nodes in the graph.

Figure 4.6 demonstrates an example used to find the ***number of paths*** and the ***length of the shortest path*** (the number of links which make up the shortest path) between the nodes (a) and (c). The highlighted cell in the adjacency matrix ($\boldsymbol{A}$) indicates that there is no single link path between them. Similarly, the ($\boldsymbol{A^2}$) shows there is no path of two links connecting (a) and (c). Therefore, the multiplication is repeated to get ($\boldsymbol{A^3}$), which presents a single path of the length of three links which joins (a) and (c). That means the total count of paths of the maximum length equals to three links is only one path and the length of the shortest path is three. The above theorem is exploited to compute the nodes which have an optimal closeness and connectivity to all other nodes of the graph (network) as following:

1. Multiply the adjacency matrix (A) by itself to find the shortest path and the number of paths between every pair of nodes. This operation stops when the shortest path and the number of paths are found for all nodes of the graph.

$$f(x) == \begin{cases} \boldsymbol{A^K} = \prod_{k=1}^{K} \boldsymbol{A^k}, & \exists\ Shortest_P = \emptyset \\ Stop, & \nexists\ Shortest_P = \emptyset \end{cases} \qquad (4.3)$$

$$Where\ \boldsymbol{A}_{Shortes\,t_P} = Array\ of\ shortest\ path\ \forall\ N,$$

$$K = rank\ of\ the\ A\ at\ the\ last\ iteration\ before\ find\ the\ shortest\ path\ for\ \forall\ N.$$

2. Calculates the nodes closeness centrality which is defined as the average of the shortest path from the node to all other nodes of the network (Sabidussi, 1966) and calculated as follows:

$$Closeness_m = \left(\sum_{i=0}^{N-1} Shortest_{P_{(m,n_i)}} \right) \Big/ N-1 \qquad (4.4)$$

$$Where\ m \neq n,\ \ and\ Shortest_P = Shortest\ path\ (m,n)$$

3. Compute the node connectivity which is defined in this thesis as the average number of paths used to connect the current node to all other nodes of the graph and it is calculated as follows:

$$Connectivity_m = \left(\sum_{i=0}^{N-1} Count_{P_{(m,n_i)}} \right) \Big/ N-1 \qquad (4.5)$$

$$Where\ m \neq n,\ \ and\ Count_P = Number\ of\ paths\ (m,n)$$

4. Normalizing the $Closeness$ and Connectivity according to the Sum-to-One normalization (STO) method to unify the range of their values (Wu, 2012), as follows:

$$Normalised\ Closeness_m = NCL_m = \frac{Closeness_m}{\sum_{i=0}^{N-1} Closeness_{n_i}} \qquad (4.6)$$

$$Normalised\ Connectivity_m = NCO_m = \frac{Connectivity_m}{\sum_{i=0}^{N-1} Connectivity_{n_i}} \qquad (4.7)$$

5. Sorting the nodes according to their "closeness centrality" ascending to define the nodes of the smallest latency that are needed to communicate with all the other nodes.

6. Sorting the nodes according to the inverse of their connectivity ascending to identify the nodes of the highest reliability in the network.

7. Creating a new weight which is the combination of the node closeness value (their latency) (ω_L) with the inverse of node connectivity value (their reliability) (ω_R) to find the final sort of the nodes, as the following:

$$Comb.NCL - NCO_n = (NCL_n \times \omega_L) + \left(\frac{1}{NCO_n} \times \omega_R \right) \qquad (4.8)$$

$$Where\ \ n \in N,\ \ \forall n = (1, \ldots., N-1).$$

8. Finally, sorting the nodes according to the new weight ascending and select the Controllers nodes (C) according to the number of controllers which are found from the Equations (4.1) or (4.2).

To summarise and simplify the above steps, they are represented by the Pseudocode which is displayed in Figure 4.7.

(G: GRAPH, NODE= N, Links=L,)
#***Initialise*** #
1. **Input**:
$Controller\ capacity\ as\ number\ of\ switches\ = \sigma^C;$
$Controller\ capacity\ as\ requests\ per\ second\ = \rho^C;$
$Number\ of\ switches\ = n;$
$Load\ matrix\ = Ld^N;$
$Conectivity\ matrix\ = AdjD;$
$weight\ of\ the\ node\ latency\ = \omega_L;$
$weight\ of\ the\ node\ reliability\ = \omega_R;$

###***Find Controllers Count*** ###
2. *Controller Number=C* $= N/\sigma^C$ *or* $= (\sum_{i=0}^{N} Ld_i^N)/\rho^C$

###***Find Controllers Locations*** ###
3. **Initialise**: $\boldsymbol{Shortest_P} = \emptyset$; */* Array of shortest paths */*
Initialise: $\boldsymbol{Count_P} = \emptyset$; */* Array of paths count*/*
While $(any\ \boldsymbol{Shortest_P}(i,j) = \emptyset)$
$\boldsymbol{A^K = A \times A^{K-1}}$;
For *all* $n \in N$ **do** *. /* for all nodes in G */*
Find shortest path and paths count by graph theorem;
Set shortest path in $\boldsymbol{Shortest_P}$;
Update path count in $\boldsymbol{Count_P}$;
End For
End While

4. **For** *all* $m \in N$ **do** *. /* for all nodes in G */*
/ Compute Nodes Closeness */*
$Closeness_m = \left(\sum_{i=0}^{N-1} Shortest_{P(m,n_i)}\right)/N-1;$
/ Compute Nodes Connectivity */*
$Connectivity_m = \left(\sum_{i=0}^{N-1} Count_{P(m,n_i)}\right)/N-1;$
Endfor

5. **For** *all* $m \in N$ **do** *. /* for all nodes in G */*
/ Normalising Nodes Closeness */*
$Normalised\ Clossness_m = \frac{Clossness_m}{\sum_{i=0}^{N-1} Clossness_{n_i}};$
/ Normalising Nodes Connectivity*/*
$Normalised\ Connectivity_m = \frac{Connectivity_m}{\sum_{i=0}^{N-1} Connectivity_{n_i}};$
Endfor

6. **Sort** according Closeness ascending (smallest latency);

7. **Sort** according inverse of Connectivity ascending (highest connectivity);

8. **For** *all* $m \in N$ **do** *. /* for all nodes in G */*
/ Combine Nodes Closeness and inverse of connectivity values*/*
$Comb.NCL - NCO_n = (NCL_n \times \omega_L) + \left(\frac{1}{NCO_n} \times \omega_R\right);$
Endfor
Sort according combined weight ascending;
Select ***C*** controllers' nodes;

Figure 4.7: The Pseudocode of part 1 of COVN placement algorithm.

The above steps find the nodes which have the optimal latency and reliability to host the physical controllers of SD-WAN.

4.7.4. The Computational Complexity of COVN Placement Algorithm Part-One

The highest computational complexity for part one of COVN placement is $\boldsymbol{O(KN^2)}$, *where* $\boldsymbol{K}$ *is the maximum iteration until all the shortest paths are founded and N is number of the network nodes.* While finding the optimal placement by brute-force search produces a large complexity because it grows exponentially when the number of controllers are increased (Heller *et al.*, 2012), according to the following equation; $All\ possibilites = \left(\frac{N!}{C!(N-C)!}\right)$, *where C is the controllers' number*. In addition, brute-force search needs to apply the shortest path algorithm for each possibility. The computing complexity of calculating the shortest path using Dijkstra's algorithm is $\boldsymbol{O(NlogN + L)}$ (Kaibel and Peinhardt, 2006). Therefore, the total computation complexity for finding controller placement using all possibilities is $\boldsymbol{O\left((NlogN + L) \times \left(\frac{N!}{C!(N-C)!}\right)\right)}$, which is much higher than the complexity of COVN placement algorithm.

4.8. The COVN Placement Algorithm Part-Two

Part-two of COVN placement algorithm places the virtual controllers for every VN separately. The placement of virtual controllers consists of two sections which are, clustering the VN and assigning the virtual controllers of this VN. In the first section, every VN is partitioned into balanced and position related clusters. The second section determines the most appropriate physical controller that could host the virtual controller of every cluster according to the switch-controller latency. The clustering of the VN is performed by a novel clustering algorithm proposed by this research, called **Peripheral Clustering Algorithm**. What follows next is the explanation of the two sections of the placement of the virtual controllers.

4.8.1. The Motivation to Propose a Novel Clustering Algorithm

Part two of COVN placement algorithm needs to partition the network into balanced, connected and converged clusters. As well as this, it is required that every VN dynamically segmented which necessitates the smallest possible computational complexity for segmentation process. Therefore, a comprehensive literature review is performed on the clustering methods to search for the demand clustering algorithm (see Section "**2.4.6.** Summary and Conclusion of Clustering Algorithms"). The literature review discloses that the clustering methods are trading off between results accuracy and computational complexity. Furthermore, the literature review could not identify a clustering method that slices the network according to position, load and connectivity with simple computational complexity.

In addition, several practical tests were conducted in this research to employ the K-means, Breadth-First Search (BFS), and Spectral Geographical Clustering (SGC) to partition the SD-WAN but they did not produce acceptable results. For instance, K-Means has a high computational complexity and generates a different shape of cluster every execution due to the random starting of the clusters' centroid. The BFS creates unbalanced clusters, and SGC can only partition the network into an even number of clusters.

The need for clustering accurately according to the pre-mentioned requirements with the lowest computational complexity was the motivation to develop a novel clustering algorithm, called Peripheral Clustering Algorithm. This new algorithm provides any number of connected clusters with a balanced load and converged positions. Also, it

consumes an accepted computation complexity, which allows it to be applied dynamically, as shown below:

$$Computation\ Complexity = Q * \sum_{I=0}^{\sigma^{Q}-1} (I) \qquad (4.9)$$

where Q: No. of clusters , σ^{Q}: Capacity of cluster .

4.8.2. The Design of the Peripheral Clustering Algorithm

As mentioned above, the first section of placing the virtual controllers is represented by partitioning the virtual network using the peripheral clustering algorithm. This algorithm could be summarised in three stages, ***Initialising***, ***clustering*** and ***balancing*** (see Figure 4.8).

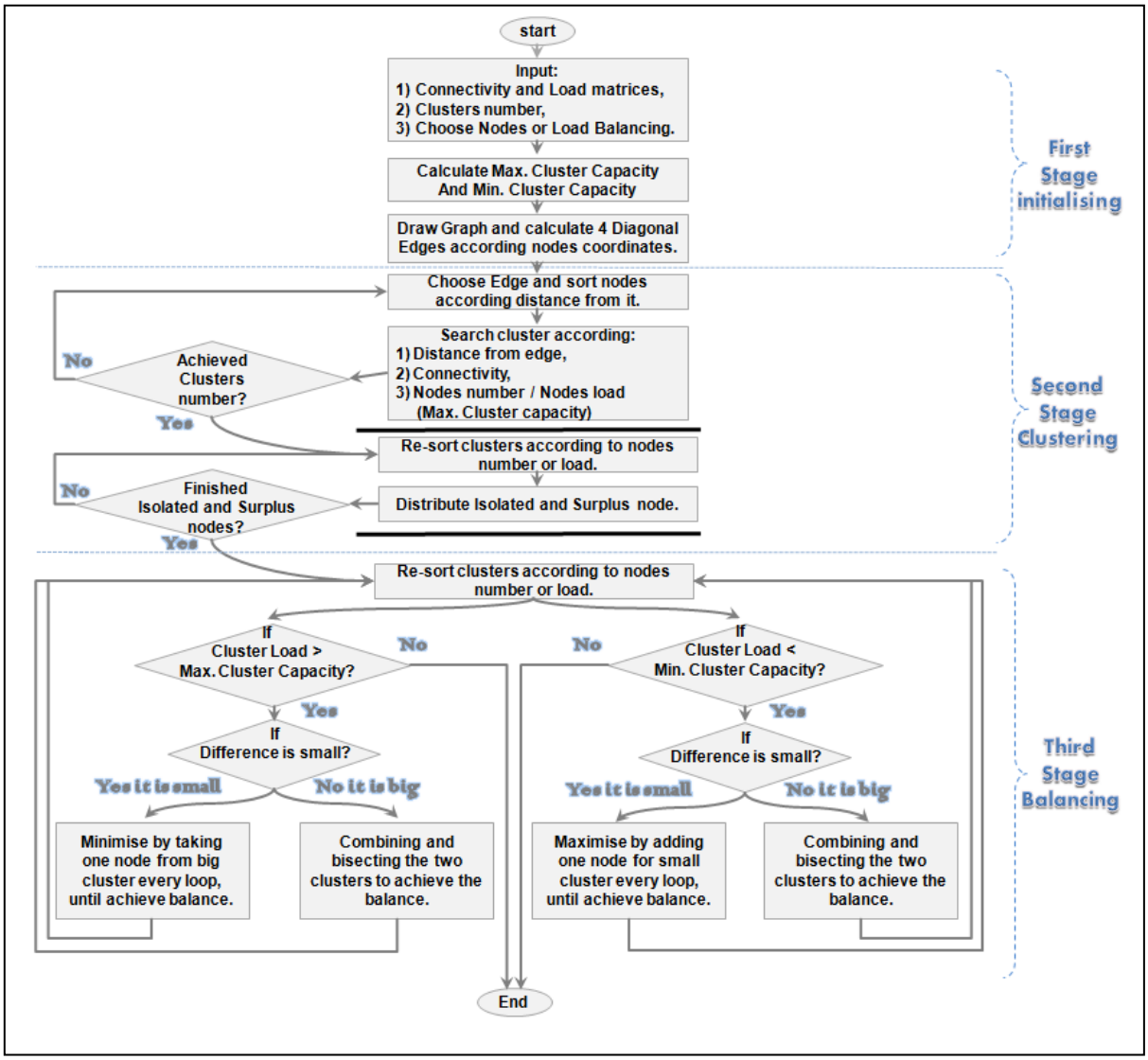


Figure 4.8: The structure of the peripheral clustering algorithm.

These three stages are explained in detail below:

A) Initialization stage: it includes: the reading of the adjacency matrix which represents the network connectivity; Also, the reading of the load matrix which identifies the load of each node if required; Furthermore, the choosing of the desired balance approach, which could be the summation of cluster nodes or the summation of their load; Followed by calculating the maximum and minimum capacity of the cluster to be used as a boundary for the cluster sizes. The initial stage is finished by drawing the graph and finding four diagonal edges for it, which would be used as a base point in the clustering process (see Figure 4.9 below).

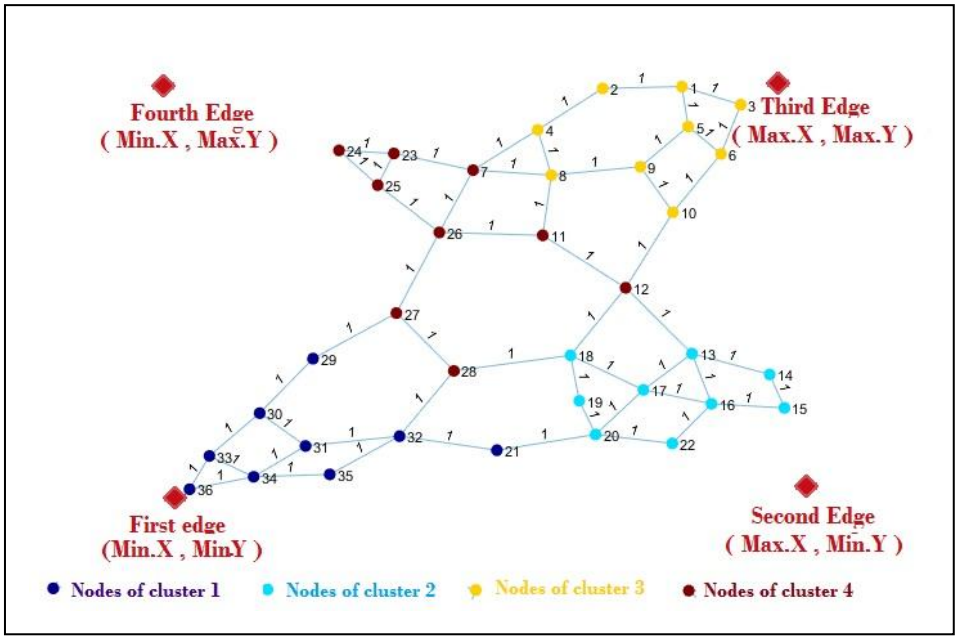


Figure 4.9: Example of the four diagonal edges, which are extracted from node coordinates.

The four diagonal edges are virtual nodes in the coordinate space and not a node of the graph. The X and Y of the diagonal edges extracted from the coordinates (X and Y) of all the nodes of the graph are as the follows:

1. Finding the Maximum and Minimum value of X for all graph nodes:

$$Max.X = \underset{n \in N}{\text{Max}}(x_1, \ldots\ldots, x_n) \qquad (4.10)$$

$$Min.X = \underset{n \in N}{\text{Min}}(x_1, \ldots\ldots, x_n) \qquad (4.11)$$

2. Finding the Maximum and Minimum value of Y for all graph nodes:

$$Max.Y = \underset{n \in N}{\text{Max}}(y_1, \ldots\ldots, y_n) \qquad (4.12)$$

$$Min.Y = \underset{n \in N}{\text{Min}}(y_1, \ldots\ldots, y_n) \qquad (4.13)$$

3. The coordinates of the first edge:

$$First\ edge = (Min.X, Min.Y) \qquad (4.14)$$

4. The coordinates of the second edge:

$$Second\ edge = (Max.X, Min.Y) \qquad (4.15)$$

5. The coordinates of the third edge:

$$Third\ edge\ = (Max.X, Max.Y) \qquad (4.16)$$

6. The coordinates of the fourth edge:

$$Fourth\ edge\ = (Min.X, Max.Y) \qquad (4.17)$$

B) Clustering stage: the second stage is the clustering stage (see Figure 4.8). At this stage the algorithm uses the four diagonal edges cyclically to create the clusters. It starts by taking the first edge and sorting all nodes of the graph according to the distance from the first edge. The algorithm is then inserts the nearest connected node to the first cluster every loop in order to create this cluster. The loop stops inserting the nodes to cluster one when the identified cluster capacity is achieved (see Figure 4.9). The graph nodes which are combined to cluster one are removed from the searching group. After that, the process is repeated with the second, third and fourth edges cyclically to create the clusters. The process repetition is stopped when the number of the required clusters is achieved or the process comes to the end of connected nodes.

In the clustering stage, some nodes are arrested among the clusters, which are defined as ***isolated nodes***. Also, the clustering stage may end before the distribution of all nodes of the graph if all clusters achieve the maximum capacity, which produces several ***surplus nodes*** (flooded nodes). The algorithm then distributes the nodes which did not join any cluster. These nodes include both isolated and surplus nodes which are mentioned above. The distribution process begins by sorting all clusters according to their load in descending order. Next, the cluster of minimum load receives one undistributed node only if any of these nodes is connected to this cluster. The algorithm then repeats the cluster sorting according to the new load after the minimum cluster receives one of the undistributed nodes. This loop continues to run until all

undistributed nodes are distributed. The distribution of isolated and surplus nodes does not demolish the cluster balance because there are only a few nodes and they are always added to the clusters with the lowest load (see Figure 4.8).

C) Balancing stage: The third stage is responsible for the balancing of the clusters which are produced by the previous operations. The balancing is performed in two directions as shown in Figure 4.8. The left one is minimising the big clusters, which performs the most balancing optimisation. However, some small clusters are not connected to big clusters, therefore, they do not optimised to get the desired capacity. For that reason, the right side is added to guarantee that the algorithm does not leave any small cluster unbalanced.

The balancing process is designed to work in two different mechanisms in order to keep a small computational complexity whatever the difference between the unbalanced clusters. This is because balancing by moving one node in every loop provides a good computational complexity if the difference is small. While, if the difference between unbalanced clusters is big, combining and bisecting these two clusters produces better complexity in this case. Also, it is noteworthy that, there are some important principles applied in programming these mechanisms to preserve a low computational complexity without decreasing the accuracy of this clustering algorithm. One principle is using ***the sparse array*** to represent the graph (the network). The sparse array is best to represent the graph which does not have a dense connection in term of less cells of information and less computational complexity for any operation on this array. Another significant principle is converting every cluster into a ***single graph object*** that has the defined external edges connect it to the other clusters. Considering the cluster as one object shortens the process of searching its connectivity because the search will include only the external edges instead of all nodes of the cluster. However, this operation increased the program difficulty because it needs to find the external edges and to update them with every change to the cluster. For more details see the pseudocode of the left side of the third stage with its explanation in the rest of this section.

The pseudocode in Figure 4.10, shows that the load could be the number of switches or the summation of their load which is represented by the flow requests per seconds. Then it finds the cluster capacity which is the average of the network load. Subsequently, the algorithm needs to calculate the balancing tolerance value which is based on the

cluster capacity and the ***ratio of balancing tolerance***. This ratio depends on the network size and the sensitivity of network toward the load balancing. For example, if the ability of the controller to manage the load is not very restricted, then the network has low load sensitivity. ***This research considers the network is sensitive to the load balance and uses the ratio equals to 0.1 in all conducted tests***. Consequently, the maximum and minimum cluster capacity is computed to be used as a boundary when constricting the clusters.

```
(G: GRAPH, NODE= N, Links=L,)
      #Initialise #
 1.   Load = switches number or summation of switches load (request per second);
 2.   Cluster Capacity = Load / No. of controllers;
 3.   Balance tolerance = (0.1 to 0.5) × Cluster Capacity;
 4.   Max. Cluster Capacity = Cluster Capacity + Balance tolerance;
 5.   Min. Cluster Capacity = Cluster Capacity;

      ### Minimising the Big Cluster ###
 6.   Set External edges;
 7.   Sort Clusters descending (max. at top)
 8.   While Cluster Load > Max. Cluster Capacity do
 9.        Sort Clusters descending (max. at top)
 10.       find Max cluster;
 11.       if [(Load. Max. cluster) − (Cluster Capacity)] > [ ( 0.1 to 0.4) × (Cluster Capacity) ]
                /* if difference is big */
 12.            find min-connected cluster and connections-pairs;
 13.            Sort connections-pairs; /*farthest from centre first*/
 14.            Check Cut-vertex;
 15.            Move node from big-to-small cluster and Update Matrices;
 16.       Elseif
              /* if difference is small */
 17.            find min-connected cluster;
 18.            Combine clusters;
 19.            Bisect the merged cluster into two clusters;
 20.       Endif
 21.  End While
```

Figure 4.10: The pseudocode of left side in the third stage of the peripheral clustering algorithm.

A next worthy step at point 6, is extracting the external edge to deal with the cluster as a single object. Then, a balancing loop is started with two different conditions. On the one hand (point 11), if the difference between the cluster capacity and load of the constructed cluster is small then move one node from the largest to the smallest connected cluster. On the other hand (point 16), if the difference is large then combine and bisect the largest and the smallest connected clusters. Identifying the difference amount as big or small depends on the ***difference weight*** which should be specified before running the algorithm (see Figure 4.10, point 16). ***The different weights are***

tested in a range of values between the (0.1 to 0.5). The results demonstrated that good execution time is achieved at weight 0.4 for most size of SD-WAN's.

Steps 10 and 12 find the maximum and minimum connected clusters and determine the pair of nodes which connect them. Then step 13, sorts the pairs of connections according to their geographical remoteness from the centre of the maximum cluster because the pair which is near the cluster centre has a higher probability of being a cut-vertex. Therefore, this step tries to minimise the iteration of finding the non-cut-vertex edge which is performed in step 14. This is because the cut-vertex edge cannot be moved from the biggest to the smallest cluster as it splits the biggest cluster into two detected clusters.

4.8.3. The Design of Assigning the Virtual Controllers

The second section is devoted to explaining how every cluster is assigned to the proper controller. The proper controller is represented by the physical controller which can host the virtual controller and have the smallest average latency to the cluster switches. In the present study, the researcher came up with a new method to allocate the controllers for all clusters. This method is called Farthest-Cluster to Nearest-Controller (the researcher refers to as FCNC for short), this explained further below:

1. The ***shortest paths array*** (number of hops) is obtained from the part one of COVN placement algorithm- which is the placement of physical controllers, the first step of 'multiply the adjacency matrix'.
2. The ***physical controller's array*** is also available from the output of the part one of COVN placement algorithm.
3. Calculating the Average Shortest Path (**Avg.ShP**) between every cluster (**Clu**) and all controllers (**Ctr**) according to the following equation:

$$\boldsymbol{Avg.ShP\ Clu_i - Ctr_j = \sum_{n=0}^{N-1} ShP_n \ \Big/ N-1} \qquad (\mathbf{4.18})$$

$$\boldsymbol{Where\ \ i \in I}: the\ clusters\ \ Index, \qquad \boldsymbol{j \in J}: the\ \ controllers\ Index,$$
$$\boldsymbol{n \in N}: nodes\ of\ \boldsymbol{Clu_i},\ \ and\ \boldsymbol{ShP}: shortest\ path\ between\ node\ and\ \boldsymbol{Ctr_j}.$$

4. Sorting all **Avg.ShP** in the table that has (J) rows of controllers and (I) columns of clusters, adding to them the nearest controller column which will be used in next steps. (see Table 4.3).

	Controller1 (Node12)	Controller2 (Node18)	Controller3 (Node28)	Controller4 (Node11)	Nearest Controller (NC)
Cluster1 (Blue)	4.5	3.5	***2.5***	4.7	2.5
Cluster2 (Turquoise)	2.1	***1.8***	2.8	3.1	1.8
Cluster3 (Orange)	***2.6***	3.6	4.4	***2.6***	***2.6***
Cluster4 (Brown)	2.4	3	2.4	***1.8***	1.8

Table 4.3: An example of sorting the average shortest path (average number of hops) between clusters and controllers (the values in table belongs to the SD-WAN in Figure 4.11 below).

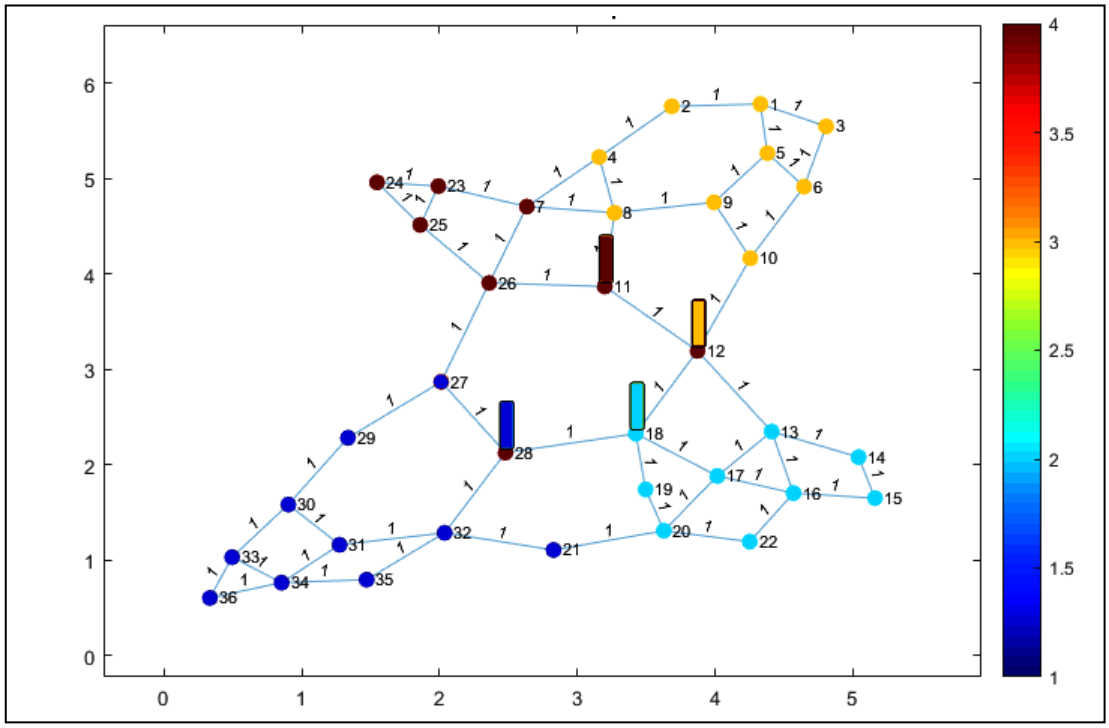


Figure 4.11: An example to explain the process of assigning the virtual controllers to clusters.

5. Identifying the smallest **Avg.ShP** for every row and set them in the new column named ***Nearest Controller*** (***NC***) in order to find the nearest controller for every cluster. The smallest values of each row in bold and underlined in Table 4.3 above.

6. Determining the biggest value in the column (***NC***) to make sure that the farthest cluster will select its nearest controller. In this example, the farthest cluster is cluster 3 (orange nodes), and its nearest controller is controller 1 (node 12). The biggest value of (***NC***) in bold and double underlined in Table 4.3 (see Figure 4.11).

7. Removing the row and column that shows interest in the selected cluster-controller value as shown in Table 4.4.

	Controller1 (Node12)	Controller2 (Node18)	Controller3 (Node28)	Controller4 (Node11)	Nearest Controller (NC)
Cluster1 (Blue)	Non	3.5	***2.5***	4.7	***2.5***
Cluster2 (Turquoise)	Non	***1.8***	2.8	3.1	1.8
Cluster3 (Orange)	Non	Non	Non	Non	Non
Cluster4 (Brown)	Non	3	2.4	***1.8***	1.8

Table 4.4: The second step of assigning the controller to cluster.

8. The second step is the reparation of the first step. It identifies the next smallest **Avg.ShP** after deleting the row and column of the first step and sets them in the **NC** column (see Table 4.4).

9. Again, finding the biggest value in ***NC*** column to let the next farthest cluster chose the nearest controller, which is cluster1 (Blue nodes) and controller3 (Node28) in this example as displayed in Table 4.4 (see Figure 4.11 above).

10. Step two ends by removing the chosen row and column similarly to the first step.

11. The above process stops when all clusters receive controllers as demonstrated in the SD-WAN in Figure 4.11 above.

The FCNC method showed that it can always minimise the worst-case latency by allowing the farthest cluster to choose its proper controller.

4.8.4. The Computational Complexity of COVN Placement Algorithm- Part 2

To determine the final computation complexity for the part two of COVN placement algorithm, it is compulsory to define the complexity of every step of this part as illustrated in the following subsections A and B.

A) The Computation Complexity of the Peripheral Clustering Algorithm

As mentioned before the network is represented by the graph G(N, L), where N is the graph nodes which are connected by (L) links. The network needs to be partitioned into ($\boldsymbol{Q}$) clusters. Every cluster has a capacity as the number of switches ($\boldsymbol{\sigma^{Q}}$) or the load of

requests per second ($\boldsymbol{\rho^{Q}}$). Each node has the degree ($\mathbf{d}$), which indicates the number of edges connected to this node. The summation of the degrees of all nodes is denoted as ($\boldsymbol{D}$). The connectivity between the graph nodes is represented by the ***Spare matrix*** ($\boldsymbol{S}$). Finally, the length of spare matrix is equal to the number graph edges also equivalent to the half of D (MATLAB, 2017).

The peripheral clustering algorithm has three main stages, initialising, clustering and balancing (see Section "4.8.2. The Design of the Peripheral Clustering Algorithm"). The computation complexity of the initialising stage is simple because it is just reading the input matrices and draw the graph. Therefore, the highest complexity in the initialising stage is as below:

Computation complexity of reading S

$$\frac{1}{2} * D_N = \frac{1}{2} * \sum_{\boldsymbol{I=0}}^{\boldsymbol{N-1}} \mathbf{d_I} \qquad (4.19)$$

$where\ \boldsymbol{D_N}: summation\ \ of\ nodes\ \ degree\ ,\ \boldsymbol{N}: No.of\ \ nodes\ ,\ \ and\ \ \boldsymbol{d}: degree\ .of\ \ node\ .$

The second stage (clustering) has three sequential steps. Step one, is sorting the nodes according to the distance from the cluster edge. Step two, is inserting the nodes into the cluster according to their connectivity. While, the computational complexity of the third step is negligible because it only distributes a few disjoint nodes to clusters.
The computational complexity of step one is:

Sorting Computational Complexity (SCC)

$$SCC = \sum_{\boldsymbol{I=0}}^{\boldsymbol{Q-1}} (\boldsymbol{N} - \boldsymbol{I\sigma^{Q}}) \qquad (4.20)$$

$where\ \ \boldsymbol{N}: No.of\ \ nodes\ ,\ \ \boldsymbol{Q}: No.of\ \ clusters\ ,\ \ \boldsymbol{\sigma^{Q}}: Capacity\ \ of\ \ cluster\ .$

While the computational complexity of step two is:

Inserting Computational Complexity (ICC)

$$ICC = Q * \sum_{\boldsymbol{I=0}}^{\boldsymbol{\sigma^{Q}-1}} (\boldsymbol{I}) \qquad (4.21)$$

$where\ \ \ \boldsymbol{Q}: No.of\ \ clusters\ ,\ \ \boldsymbol{\sigma^{Q}}: Capacity\ \ of\ \ cluster\ .$

The third stage (balancing) has a smaller computational complexity than the previous two stages because it deals only with the unbalanced clusters. The unbalanced clusters are a small group. Furthermore, the stage consideres the cluster as a single object and defines its connectivity using external edges which minimise the searching of its

connected neighbour clusters. Therefore, its computational complexity is small and negligible as well.

B) The Computation Complexity of Assigning Virtual controllers

The process of assigning the controllers to the clusters depends on the number of clusters and number of controllers. The computational complexity of this process is:

Assigning *Computational Complexity (ICC)*

$$\boldsymbol{ACC = Q * C}_{where \quad \boldsymbol{Q}:No.of\ clusters\ ,\ \boldsymbol{C}:No.of\ controllers\ .} \tag{4.22}$$

Therefore, the ACC is also very small and of an ignorable complexity.

C) The Final Worst-Case Computational Complexity

From the previous equations of computational complexity, it is possible to identify that the computational complexity of step two in the second stage of peripheral clustering algorithm (see Equation 4.21) represents the greatest one. Therefore, the Equation 4.21 is considered as the ***worst-case computational complexity of part two of COVN placement algorithm***, as it is redefined below in Equation 4.23.

Worst-Case Computational Complexity

$$O\left(Q * \sum_{\boldsymbol{I=0}}^{\boldsymbol{\sigma^Q-1}} (\boldsymbol{I})\right)_{where \quad \boldsymbol{Q}:No.of\ clusters\ ,\ \boldsymbol{\sigma^Q}:Capacity\ \ of\ cluster\ .} \tag{4.23}$$

It is of relevance to mention that the cluster capacity in the previous equations could be ($\boldsymbol{\sigma^Q}$) or ($\boldsymbol{\rho^Q}$) therefore the used ($\boldsymbol{\sigma^Q}$) could be replaced by ($\boldsymbol{\rho^Q}$), if the second one is the implemented capacity in clustering.

4.9. Summary and Conclusion

The chapter displays the problem and its solution, which is represented by the COVN placement algorithm. Then, the usage and dynamic implementation of COVN algorithm are described in detail.

After that, the COVN algorithm is introduced as shown in Figure 4.12. The objectives, requirements, and purposes are defined. Then, the three scenarios are investigated in order to show the validity of the method which is used in the COVN placement

algorithm. It is important to note that the used method differs from all the previous placement methods where the controller placement considers all SD-WAN as a single domain and does not depend on the set of nodes (cluster). After that, the design of COVN algorithm is demonstrated as two parts. Part one places the physical controllers, while part two is the placement of virtual controllers. Part two is explained in two sections which are the clustering by the novel peripheral clustering algorithm and assigning the virtual controllers to the resulting clusters (see Figure 4.12).

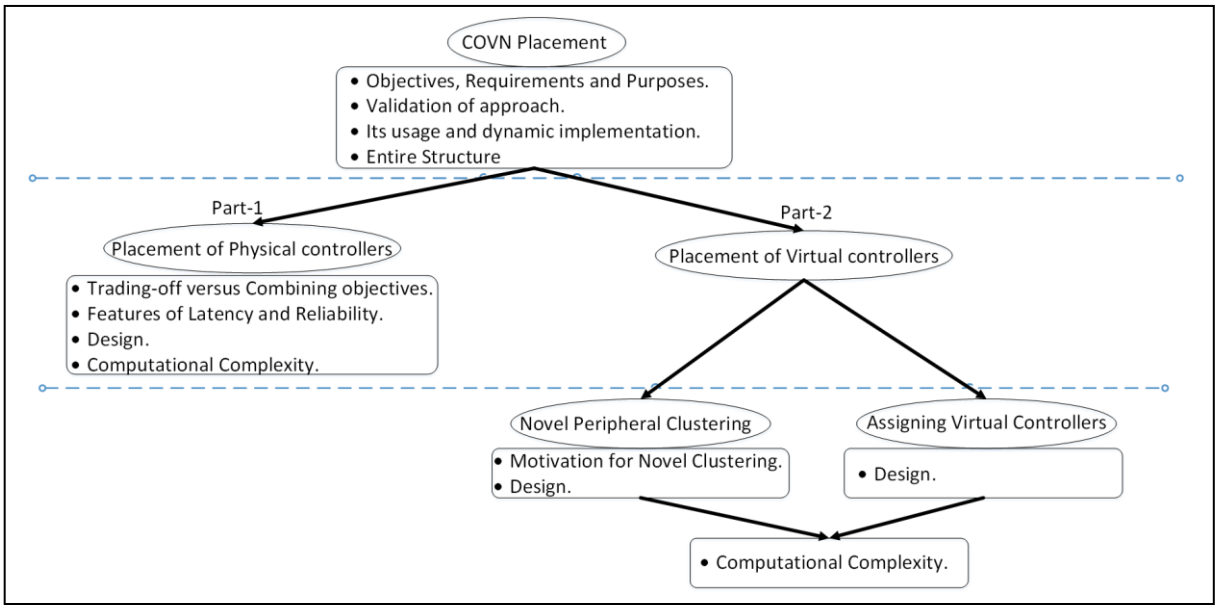


Figure 4.12: The flowchart of introducing the COVN placement algorithm in this chapter.

The computational complexity is discussed at the end of each section to show that the COVN placement algorithm has achieved a low computational complexity for both main parts. The complexity of the physical controller placement is a little higher than the complexity of the virtual controller placement. That is fitting the requirements of this model because part two is applied more frequently than the part one.

Finally, it is worthy to mention that part one used the method to find the shortest path faster than searching all possibilities of shortest paths between switches and controllers. While, part two uses geographical location to cluster the nodes instead of the path to obtain a faster clustering process. Also, part two employs the sparse matrix and constructs the cluster as a single object to simplify the balancing operation.

CHAPTER FIVE
COVN PLACEMENT TESTBEDS

5.1. Introduction

This chapter presents the two testbeds and the assisting programs which have been produced for examining the COVN placement algorithm over SD-WAN with multiple virtual networks. The first testbed is represented by the new simulator which is built by the researcher and called ***COVN simulator***. The second test platform is acted by the developed Mininet emulator. While, the assisting programs are Matlab functions, which performing the initial preparations on the topology to make it usable by the testbeds, such as extracting the distances between the nodes and converting it to Mininet topology. The chapter starts by introducing the used test platforms and the reasons for using or discarding each of them. Then, it introduces the new simulator in detail by explaining its tabs and their functionalities. After that, the Mininet emulator is presented to show its functional shortage which prevents the implementation of the controller placement. Subsequently, the proposed development of Mininet emulator is demonstrated. Further, the assisting programs are briefly demonstrated and their hardware specification is stated. Finally, the chapter ends with a summary and a conclusion.

5.2. SDN Test Platforms

The network researchers and network vendors need to test and evaluate the performance of software and hardware products before deploying it to the real network. They are using a test system called the 'test platform'. There are many types of test platforms, but the most common three are the simulators, the emulators and the online testbed (Heller, 2013). A network simulator is a software model which intends to examine the performance of the network through discrete events . The emulators mimic the behaviour of real network components and use an actual clock of the physical device which runs over it. Therefore, it produces more realistic results than the simulators (Heller, 2013). Lastly, the online testbeds are provided by the network communities to shared the access to their physical devices for testing the performance of network innovations.

The above definitions demonstrate that both the simulator and the emulator are needed to be used for implementing the COVN placement algorithm. The COVN simulator is built to compute the controller placement and to apply its decision on SD-WAN. After that, the simulator calculates the statistics of the switches-controllers latency, the connection flow-setup latency and the nodes reliability. Simultaneously, the Mininet emulator is required for additional evaluation of the implementation of these decisions and to validate the results and the statistics of the simulator.

On the contrary, the online test platforms are discarded because it is not possible to modify them to the desired requirements that are needed to implement the COVN placement algorithm. The modification is, however viable for the Matlab simulator and the Mininet.

The disadvantages that prevent the use of every testing platform are summarised in Table 5.1. Some of them are solved in order to produce a testbed for COVN placement such as creating a specific simulator for COVN algorithm and to modify Mininet to implement controller placement. While, the others cannot be sorted such as the disadvantages of the online testbeds and the general simulators.

	Online testbeds	Simulators	Emulators
Names	• Emulab. • Internet2. • GENI. • OFELIA.	• NS3 • GNS3 • Matlab POCO tools.	• Mininet
Disadvantage	1. Restricted topology. 2. Restricted technique. 3. Not replicable results.	1. General: does not support controller placement. 2. POCO: does not support another placement. 3. Low results realism. a) Not real clock. b) Not full functionality.	1. Does not support the controller placement. 2. Does not produce statistics.

Table 5.1: The disadvantages of the test platforms which prevent the implementation of COVN Placement algorithm.

5.3. COVN simulator

This section demonstrates in detail the COVN simulator which represents the Graphical User Interface (GUI) for implementing the COVN algorithm. This simulator consists of eight tabs as shown in Figure 5.1. Every tab has different functionality. The first and second tabs perform the placement of physical and virtual controllers. The third and

fourth tabs are responsible for generating the physical and virtual topology if required. The last four tabs calculate the statistics of latency and reliability in the failure-free and failure status as will be explained later.

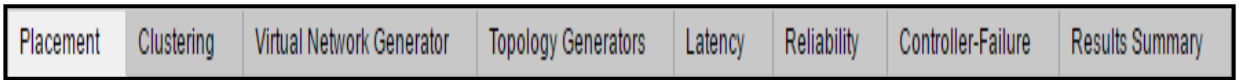


Figure 5.1: The eight tabs of COVN simulator.

5.3.1. First Tab (Placement)

This tab executes the part-one of COVN placement algorithm by computing the placement of the physical controller according to the latency (closeness) and reliability (connectivity) to all other nodes (see Section 4.7). The number of physical controllers represents the maximum required number of the active and backup controllers that are required to manage the SD-WAN according to its maximum load (number of nodes/their load).

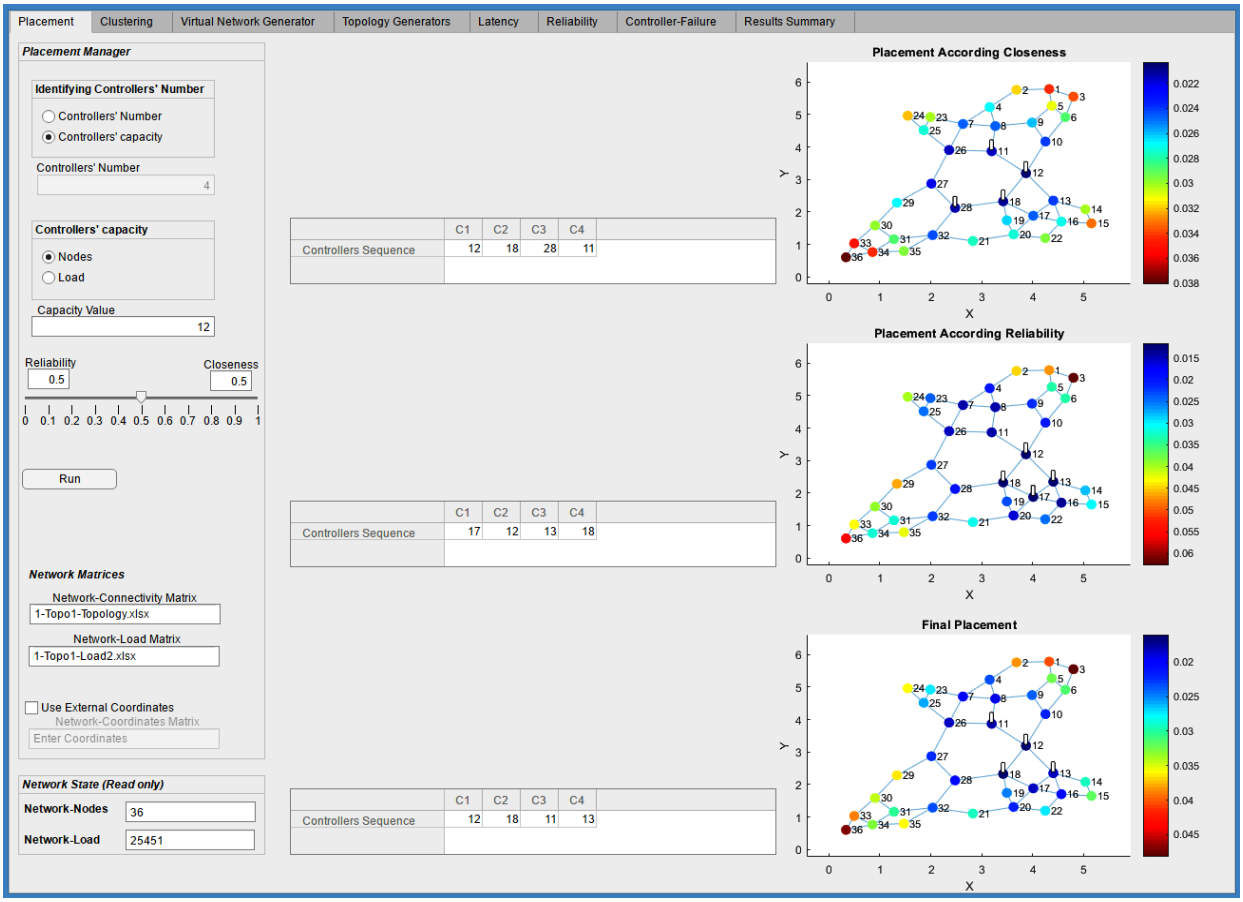


Figure 5.2: The first tab of COVN simulator (part-one of COVN placement).

The required input and parameters are entered in the panel at the left side of the tab as shown in Figure 5.2. These parameters are:

- Selecting the Controllers' Number or Controllers' capacity.

- If controller capacity is chosen, then it needs to enter the controller capacity as a number of nodes or their load (number of flow-request per second).
- Identifying the preferred weight for the latency against the reliability.
- Entering the adjacency matrix that represents the network connectivity.
- Entering the matrix of the expected load of the flow requests per second.
- Entering the coordinates of the network if it is a real topology, otherwise, the Matlab creates the coordinates of the network.

By the end of the left side, the simulator presents the number of network nodes and their load summation at initial status. The right side of the tab displays three tables and figures. The first table and figure represent the placement according to the latency (closeness) only. The second table and figure offer the placement regarding the reliability (connectivity) only. The last table and figure shows the final placement which is the placement according to the entered weight of latency/reliability.

Finally, the colour-bar shows the importance of the network nodes in relation to the used property like the closeness, the reliability or both.

5.3.2. Second Tab (Clustering)

The second tab implements the part-two of the COVN placement algorithm which is the placement of the virtual controller that is executed independently for every virtual network. This part starts by clustering the network in the light of balancing the nodes number or their load. After that, every controller is assigned to the cluster according to the minimum average latency between the controller and cluster's switches (see Section 4.8). As shown in Figure 5.3, the input parameters are similarly entered in the panel at the left side, as follows:

- Selecting the type of load (nodes or request per second).
- Entering the Clusters number or cluster capacity to find its number.
- Choosing the placement of complete network or just a virtual network.
- Entering the adjacency matrix that represents the network connectivity.
- Entering the matrix of the expected load of the flow requests per second.
- Entering the coordinates of the network if it is a real topology, otherwise, the Matlab creates coordinates of the network.

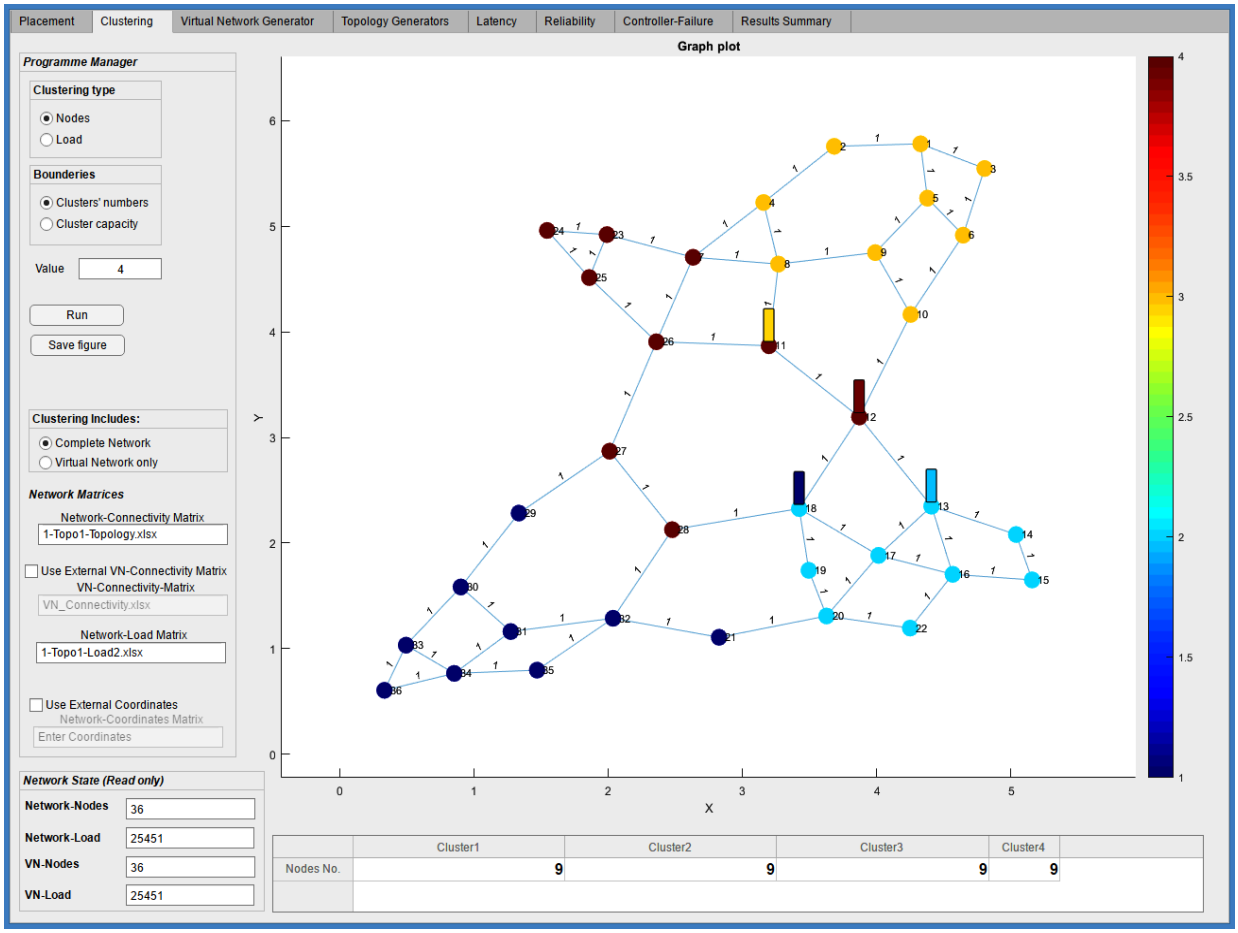


Figure 5.3: The second tab of COVN simulator (part-two of COVN placement, clustering base on nodes number).

The end of the left side of this tab presents the nodes number and their load summation for complete network and for the virtual network which is under the process of the placement. The graph plot displays the clustered SD-WAN and the controller of every cluster. The colour-bare indicates the sequence of the clusters that are created through the clustering operation.

It is important to identify that, if the clustering is performed according to the number of nodes, the table under the graph shows only the number of nodes of each cluster (see Figure 5.3 above). However, if the clustering is conducted based on the nodes load, then the table below the graph presents the number of nodes and their load summation for every cluster (see Figure 5.4 below).

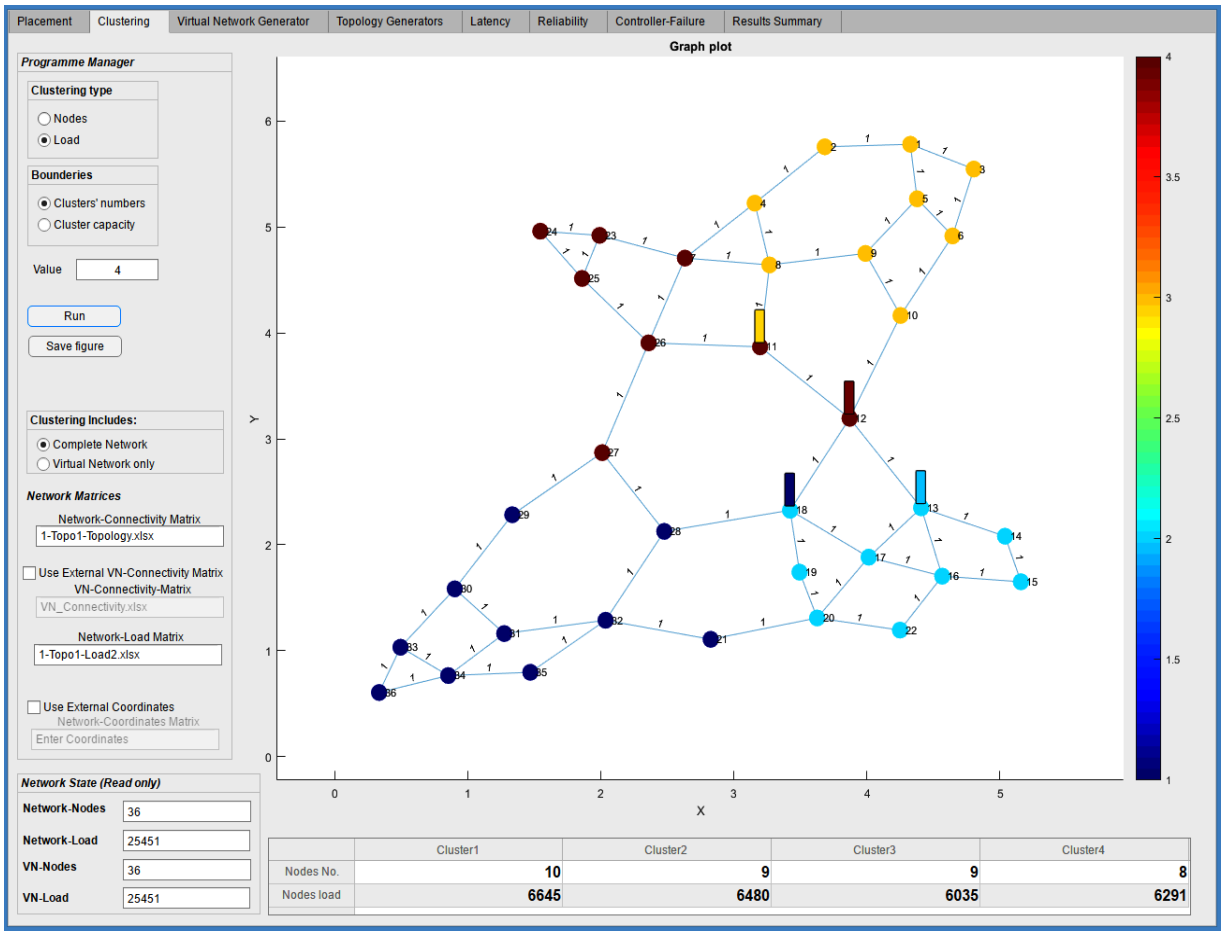


Figure 5.4: The second tab of COVN simulator (part-two of COVN placement, clustering base on nodes load).

5.3.3. Third Tab (Virtual Network Generator)

This tab creates an arbitrary virtual network or a specified virtual network according to the needs of the SD-WAN examination. The shape of the virtual network can be identified by the using the control buttons on the left side of the tab (see Figure 5.5), as follows:

1) If the Reliable VN Check-Box is selected, then the VN nodes are connected by at least two paths.

2) If the Farthest Nodes Button is chosen, then it needs to select:

a) Horizontal VN (Down or UP).

b) Vertical VN (Left or Right).

c) Diagonal VN (Front or Back).

This way produces a VN that extends in the selected direction without the needs to specify any node.

3) If the Random Nodes Button is chosen, then the nodes that are included in the virtual network need to be specified in the field of Random Nodes Parameters, which is located below the previous options. This way generates a VN that joins together the specified nodes.

4) Enter the adjacency matrix that represents the network connectivity in the field under the run button.

5) Enter the coordinates of the network if it is a real topology. Otherwise, the Matlab creates coordinates of the network.

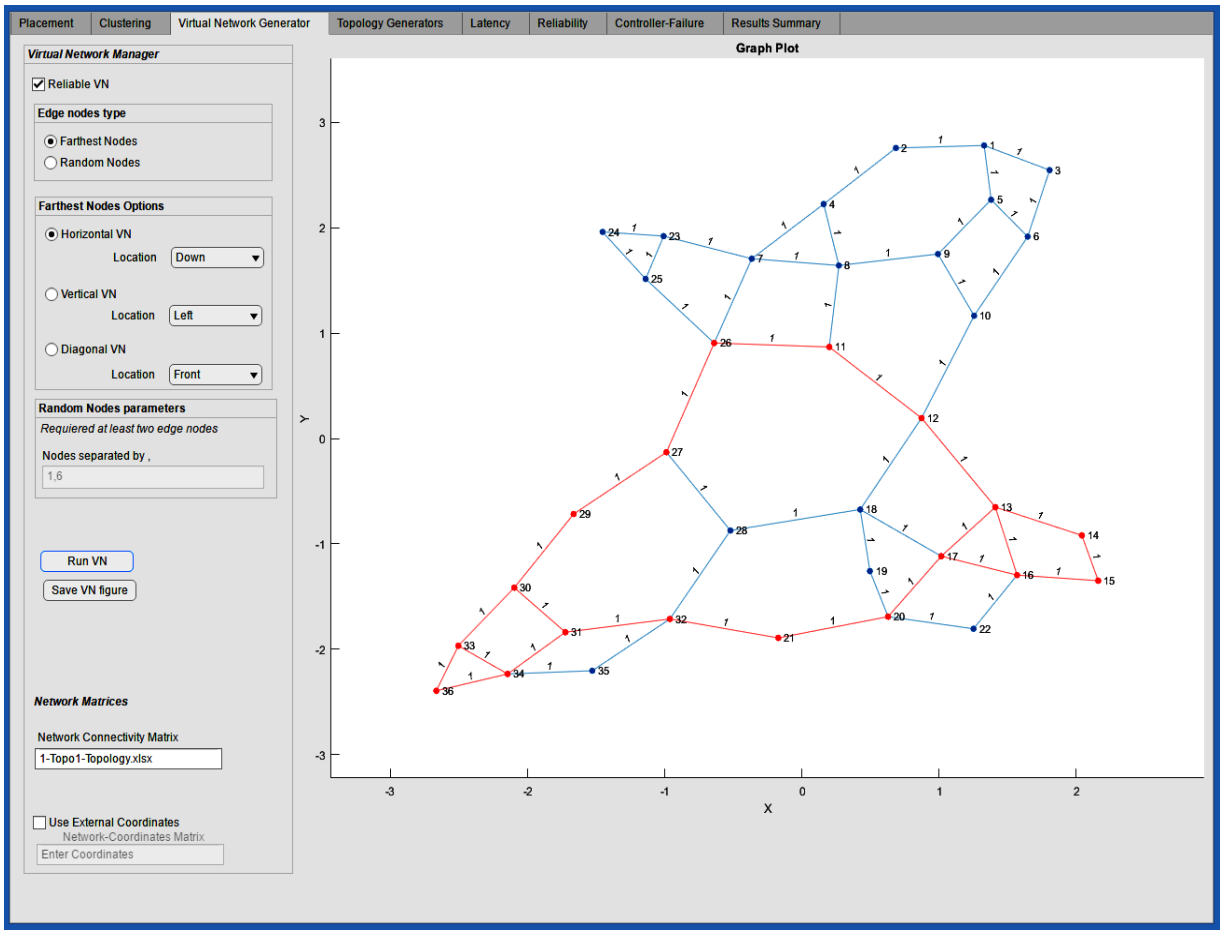


Figure 5.5: The third tab of COVN simulator (Virtual Network Generator).

5.3.4. Fourth Tab (Topology Generators)

This tab could be used to generates a topology or just create matrices for a given topology such as distance, bandwidth and load matrices. Therefore, it is very useful to produce or modify test topologies in order to examine them and analyse their behaviour in the different status. The use of this generator is displayed below with its figures.

A) Generating a new topology: this tab has two different topology generators. Every topology generator has a different method to generate a topology (see Figures 5.6 and 5.7).

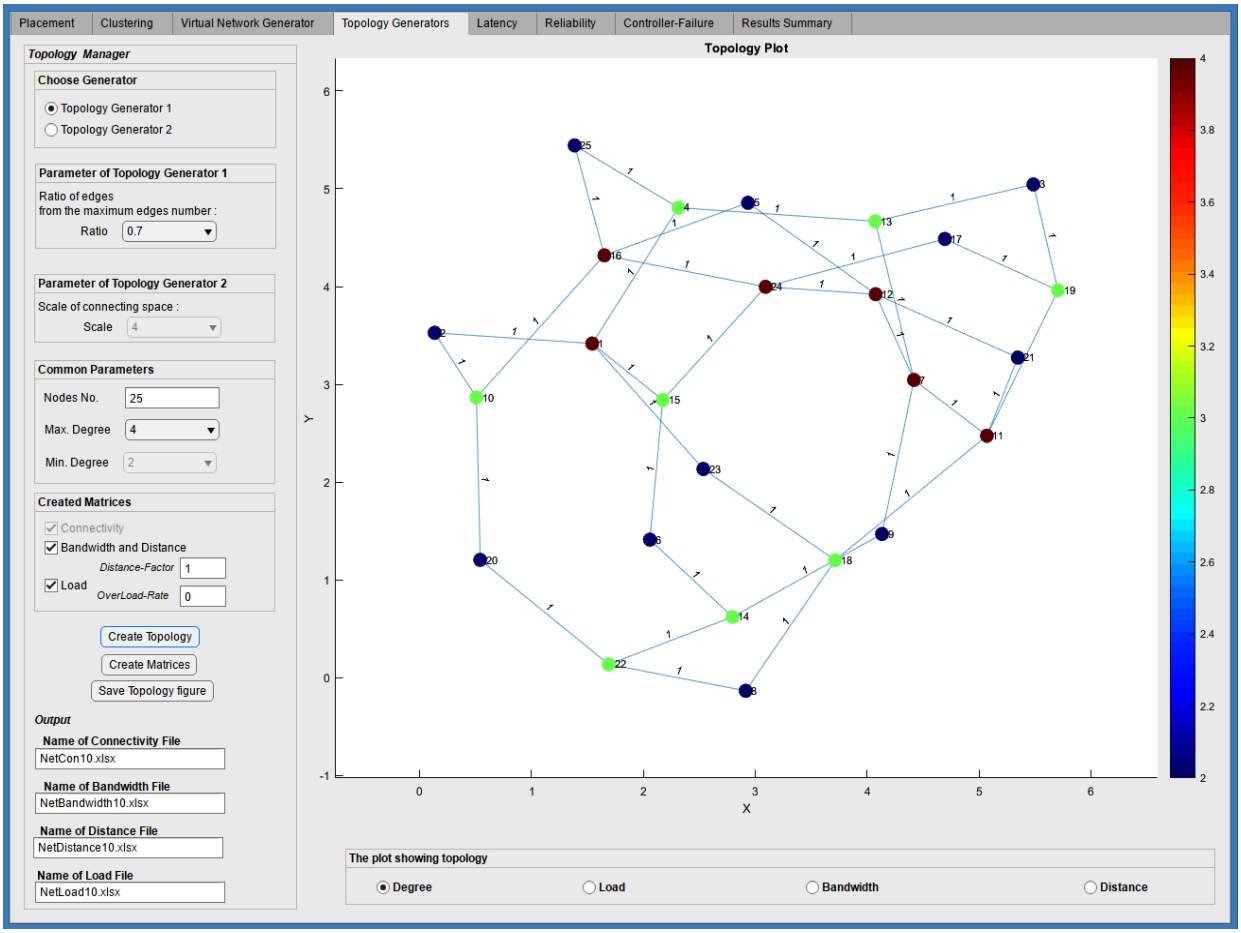


Figure 5.6: The fourth tab of COVN simulator (Topology Generator, Generator-one).

1) Generator-one, creates a network in which its nodes are connected in a completely random manner, as shown in Figure 5.6. This generator produces a network which could join the first node with the last node because it does not provide any control over its randomisation of the connectivity. This generator is good for building large-sized network, but its networks extend in just about in all directions equally.

☐ Using the Generator-one

- Chose Topology Generator 1 button.
- Enter the Edge ratio
- Enter Nodes No.
- Select Max. Degree and Min. Degree of graph nodes.
- Select Bandwidth and Distance Check-Box if their matrices are required.
- Select Load Check-Box if its matrix is required.
- Press Create Topology button.

☐ The work of Generator-one

- It starts by reading the number of nodes and edge ratio from the entered parameters. Then, it calculates the number of graph edges according to the following equation:

$$\boldsymbol{No.Edges} = \lfloor (\boldsymbol{n} \times \boldsymbol{Max.Degree}/2) \times \boldsymbol{Edges\ ratio} \rfloor \qquad (5.1)$$

- It uses the uniform distribution function to distribute the edges between the nodes of the network. The distributed function of the generator-one formulates as follows:

$$\boldsymbol{f(n)} = \left(\frac{\mathbf{1}}{\boldsymbol{Max.Degree} - \boldsymbol{Min.Degree}} \right), \quad \forall\, n \in N \qquad (5.2)$$

- It creates the selected matrices for the topology as will be explained in point (B).
- It plots the topology.

2) Generator-two, creates a network while providing some control over the connectivity randomisation in order to prevent the joining of the very far away nodes would produce a longitudinal network (see Figure 5.7 below).

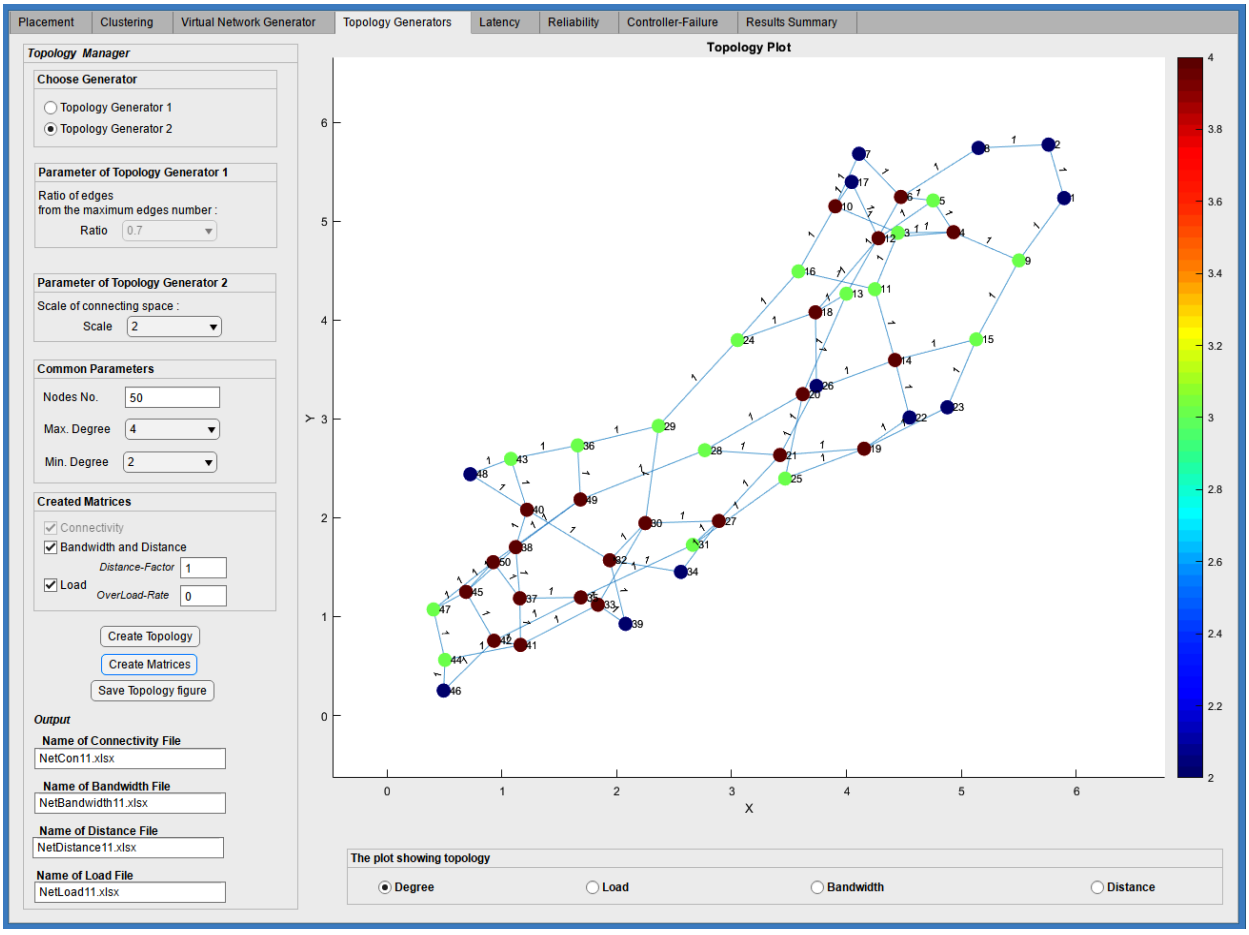


Figure 5.7: The fourth tab of COVN simulator (Topology Generator, Generator-two).

☐ Using the Generator-two

- Chose Topology Generator 2 button.
- Select the Scale to specify the connecting space.
- Enter Nodes No.
- Select Max. Degree and Min. Degree of graph nodes.
- Select Bandwidth and Distance Check-Box if their matrices are required.
- Select Load Check-Box if its matrix is required.
- Press Create Topology button.

☐ The work of Generator-two

- It begins by reading the Nodes No. and a Scale value.
- It computes the allowed searching space which could be used to find the next nodes that should be connected to the current node, according to the Equation 5.3.

$$\boldsymbol{Search\ Space = Scale \times Max.Degree} \quad (5.3)$$

- It uses the same uniform distributed function to distribute the edges onto the graph nodes (see Equation 5.2), but here, the node is only connected onto other nodes in the restricted search space, which is identified above.
- It creates the selected matrices for the topology as will be explained in point (B).
- It plots the topology.

Finally, it is worth noting that the colour-bar indicates the degree of the node in the graph.

B) Creating matrices: Additionally, this tab can create the distances and the bandwidth of the link for an existing topology. Also, it can produce the load matrix for network switches.

1) Creating the distance matrix: the distance matrix is important to examine the implementation of the controller placement algorithm. Therefore, this tab can establish a distance matrix in different ranges of distances (see Figure 5.8).

☐ Using the tab to create the distance matrix

- Enter the name of the connectivity file (adjacency matrix of graph connectivity).
- Select the Bandwidth and Distance check-box.
- Enter the value in the Distance-Factor field to scale up or down the distances.
- Press the button of Create Matrices.
- To display the bandwidths on the graph, only choose Distance-button that located in the plot showing the panel under the graph.

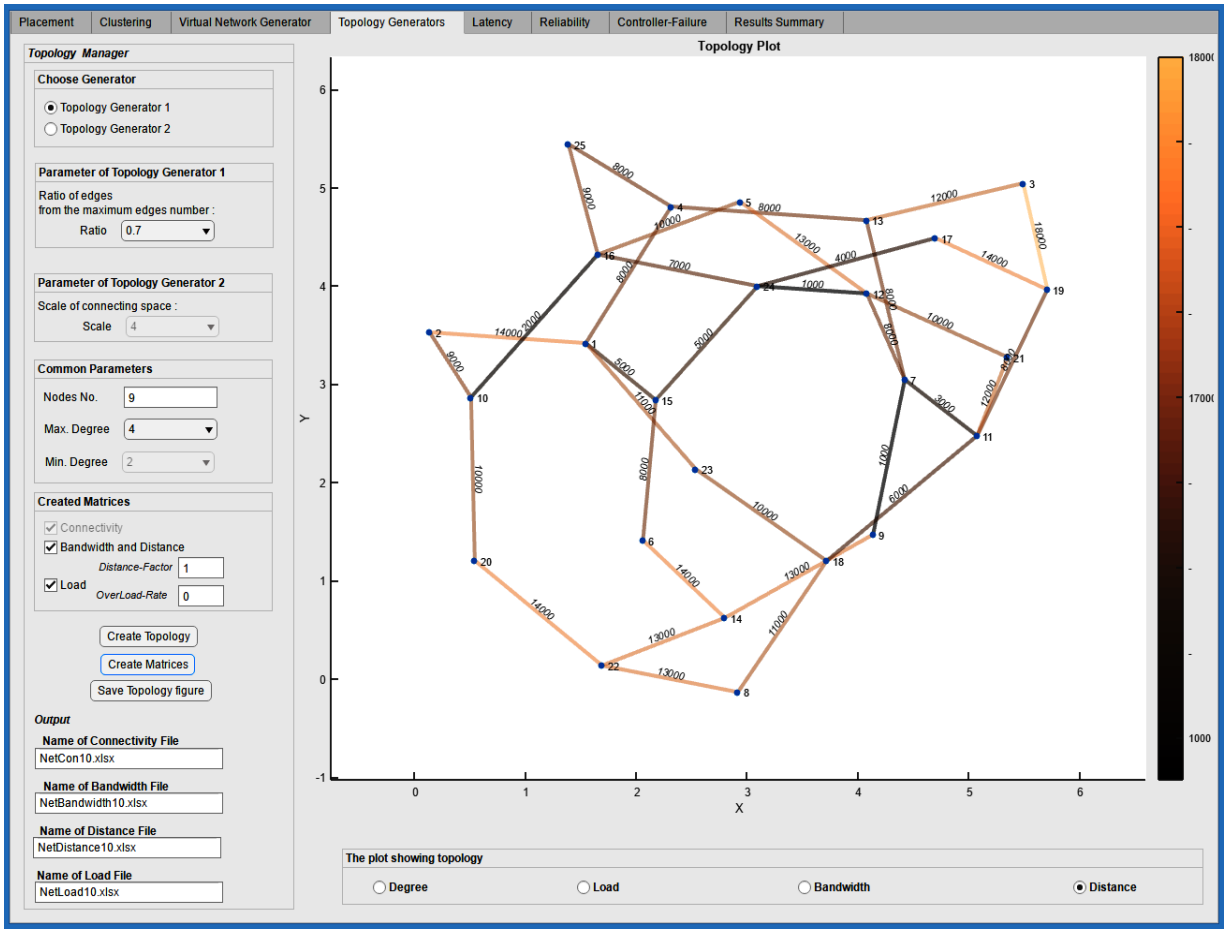


Figure 5.8: The fourth tab of COVN simulator (Topology Generator, Creating distance matrix).

☐ The working steps to create the distance matrix

- Compute the betweenness centrality (***bc***) for all nodes of a network according to the following equation (Matlab, 2017):

$$bc(n) = \sum_{s,d \neq n} \frac{No.Shp_{s,d}(n)}{No.Shp_{s,d}} \qquad (5.4)$$

$Where\ n, s, d \in N, \quad s: source\ node, \quad d: distination\ node,$
$No.Shp_{s,d}(n): number\ of\ shortest\ paths\ between\ s\ and\ dthat\ pass\ in\ node\ n,$
$No.Shp_{s,d}: the\ total\ number\ of\ shortest\ paths\ between\ s\ and\ d.$

- Divide the network nodes according to its ***bc*** into three groups (high, medium and low).
- Split the distance into three ranges (short, medium and long).
- For the nodes of high ***bc*** randomly find the distance from the first range, for the nodes of medium ***bc*** find the distance in second range and the low ***bc*** nodes use the third range.
- Amplify or diminish the distance using the Distance-Factor.
- The distances are between (1 to 20 km), which are the optimal lengths for SD-WAN links that could apply this algorithm. Also, the algorithm could achieve

good results for longer links, therefore Distance-Factor is set to scale up the length of optics fibre links. See the available length of optics fibre links in (Cisco, 2016).

2) Creating the bandwidth matrix: the bandwidth is generated using a simple method (see Figure 5.9 below).

☐ Using the tab to create a bandwidth matrix

- Enter the name of the connectivity file (adjacency matrix of graph connectivity).
- Select the Bandwidth and Distance check-box.
- Press the button 'Create Matrices'.
- Choose the Bandwidth-button which is located in the plot showing the panel under the graph to only display the bandwidths on the graph.

☐ The working steps to create the bandwidth matrix

- Compute the betweenness centrality (***bc***) for all nodes of a network according to the Equation 5.4 above:
- Divide the network nodes according to its ***bc*** into three groups (high, medium and low).
- Split the bandwidth into three ranges (high, medium and low).
- For the nodes of high ***bc*** randomly find the bandwidth from the first range, for the nodes of medium ***bc*** search bandwidth in the second range and the low ***bc*** nodes use the third range.

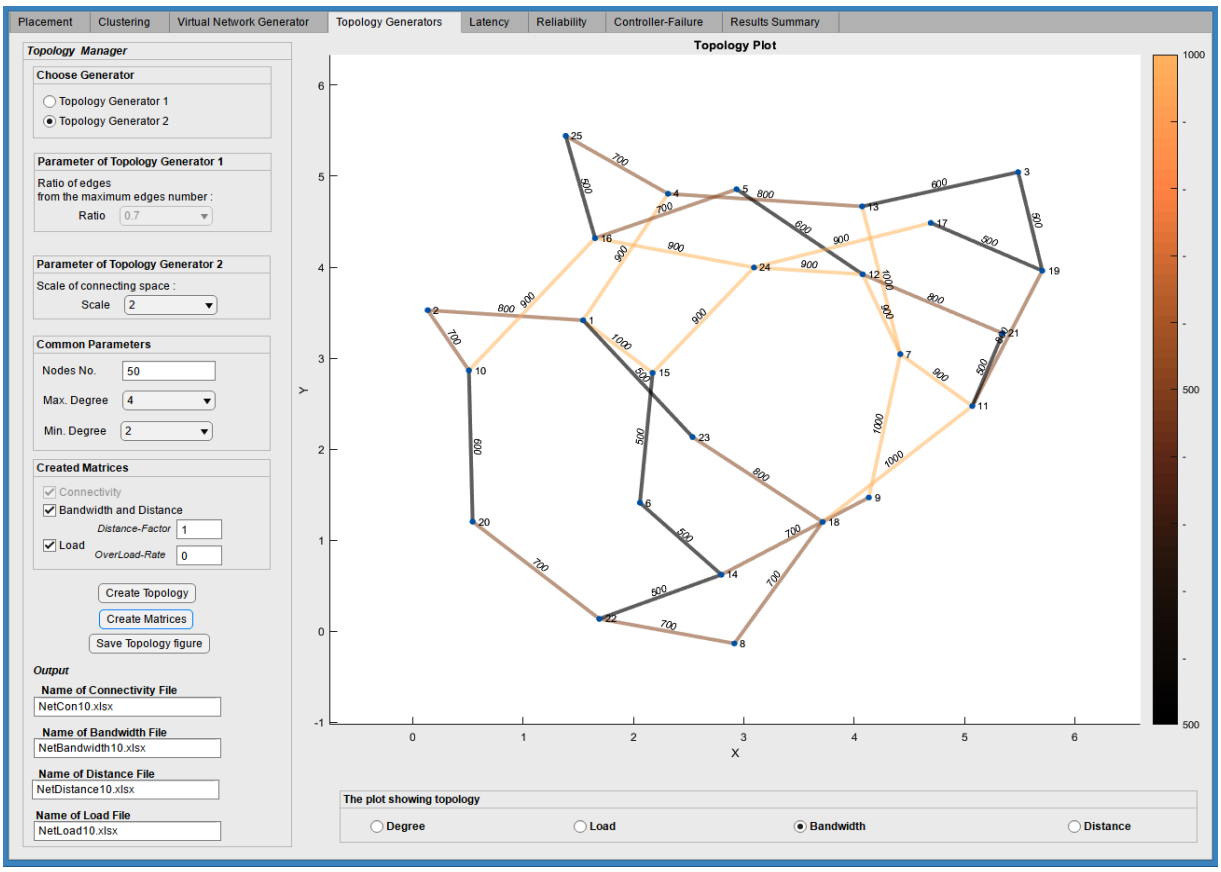


Figure 5.9:The fourth tab of COVN simulator (Topology Generator, Creating bandwidth matrix).

3) Creating the load matrix: the load matrix is needed to implement load balancing when implementing the COVN algorithm, and can be created as follows (see Figure 5.10):

☐ Using the tab to create a load matrix

- Enter the name of the connectivity file (adjacency matrix of graph connectivity).
- Select the load check-box.
- Enter the value in the Over-Load-Rate field to scale up or down the loads.
- Press the button of Create Matrices.
- To display the load on the graph, choose the Load-button which is located in the plot showing the panel under the graph.

☐ The working steps to create load matrix

- Compute the betweens centrality (***bc***) for all nodes of a network according to the Equation 5.4 above:
- Divide the network nodes according to its ***bc*** into three groups (high, medium and low).
- Split the load into three ranges (high, medium and low).
- For the nodes of high ***bc,*** randomly find the bandwidth from the first range, for the nodes of medium ***bc,*** search bandwidth in the second range and the low ***bc*** nodes use the third range.
- Amplify or diminish the loads using the Over-Load-Rate.

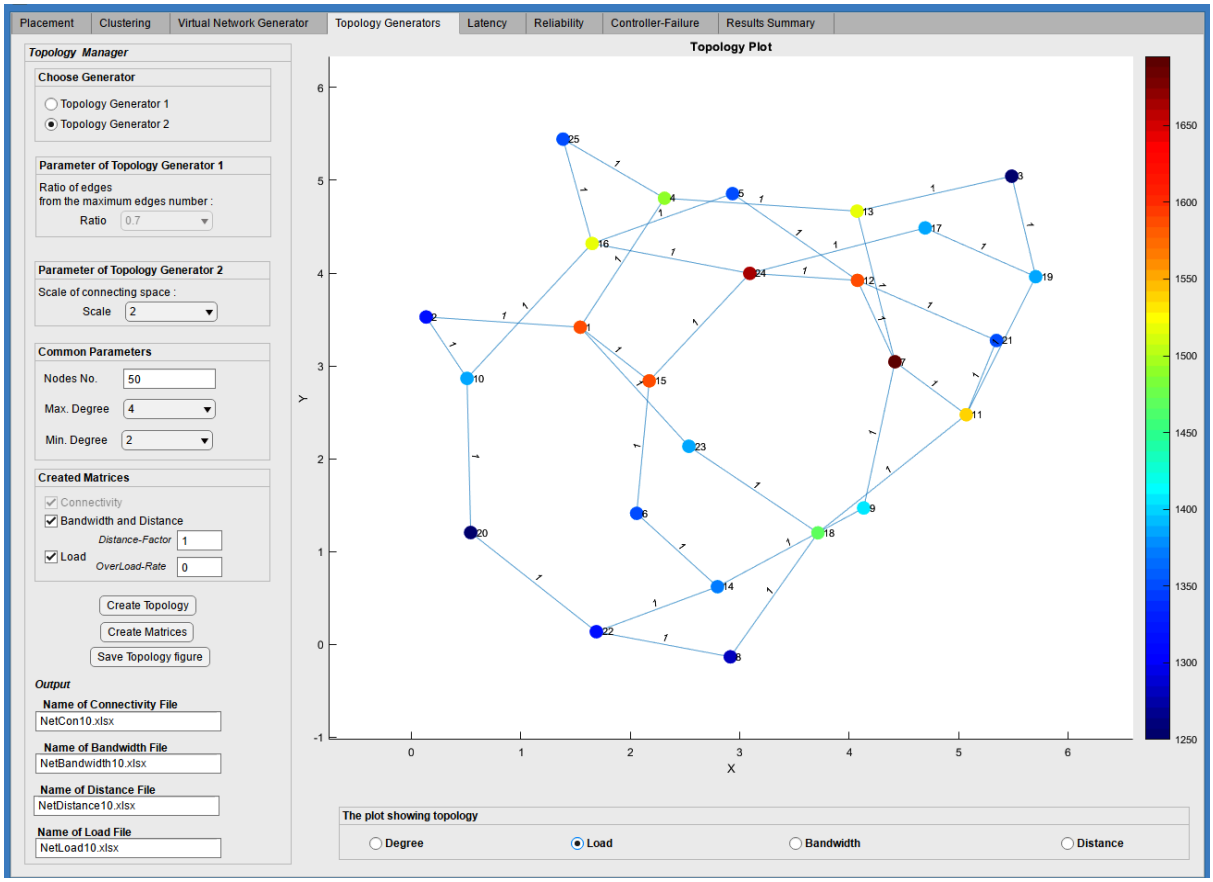


Figure 5.10: The fourth tab of COVN simulator (Topology Generator, Creating load matrix).

5.3.5. Fifth Tab (Latency)

This section is important because it presents the results of the latency that reflects the effect of implementing the COVN placement algorithm. It starts by briefly introducing the commonly used latency metrics, which is represented in the two top figures of the node-controller latency and maximum inter-controller latency. After that, the section displays the comparison of the newly added latency metrics, which are: the connection control-path latency versus the connection flow-setup latency (see Section 3.3.5).

The tab in Figure 5.11 shows the plots of all the mentioned metrics; it also displays the maximum, average and minimum values for them (see Figure 5.11). It is noting that this tab is used to find these latencies before and after controller failure (failure-free and failure statuses). The required matrices needed to calculate these latencies are the topology distances matrix and controllers-clusters matrix which is produced by the COVN placement algorithm. The used mentioned latencies are defined in the following:

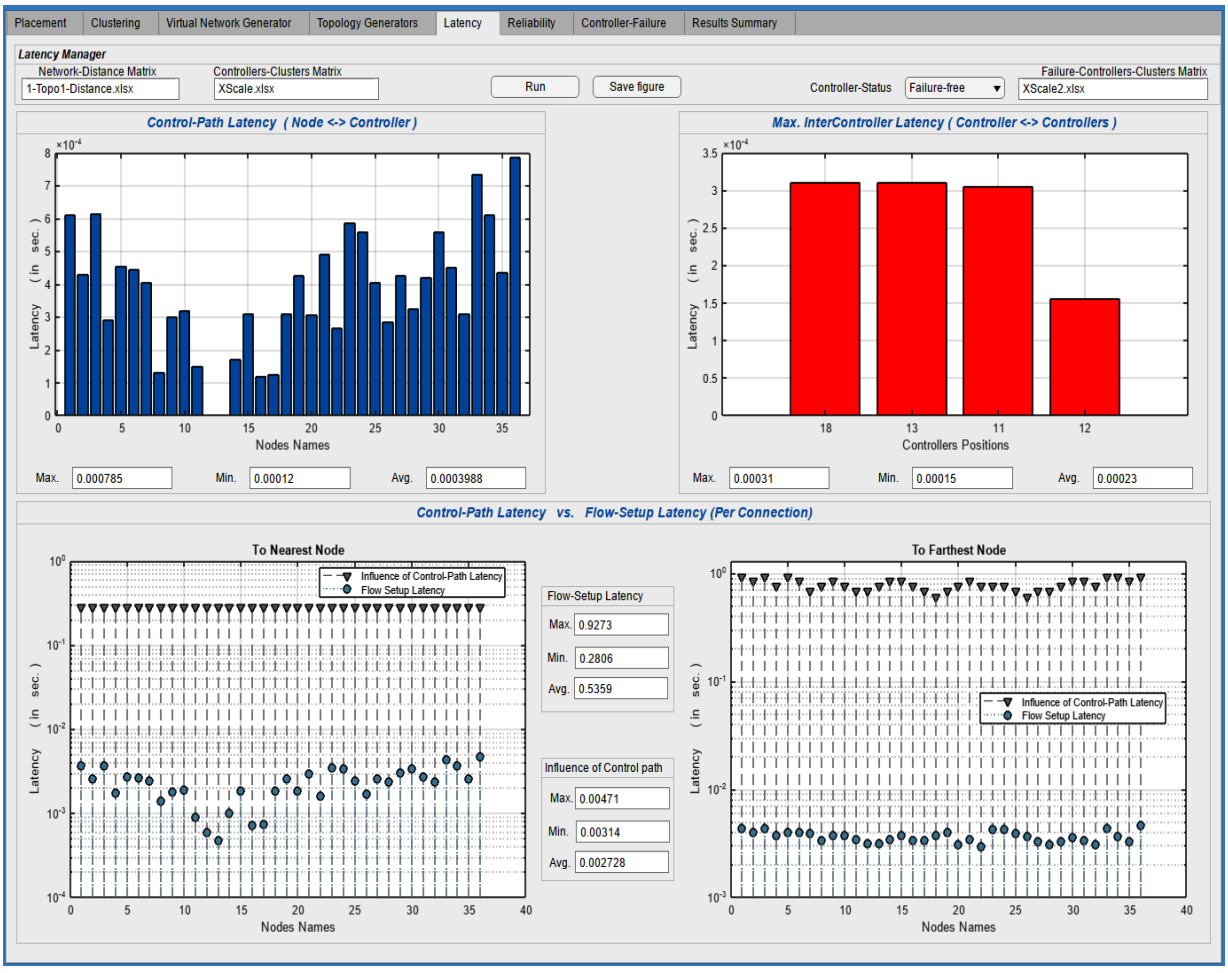


Figure 5.11: The fifth tab of COVN simulator (Latency).

A) Node-Controller Latency: It expresses the latency between every node and its controller (switch-controller latency). This latency is improved by applying the COVN

placement, therefore, it is exposed in the first figure on the upper left of this tab. This latency was used by previous researchers, and it is calculated here as well. Also, the maximum, average and minimum node-controller latency is shown under the bar-chart.

B) Maximum Inter-Controller Latency: The second bar-chart on the upper right of this tab discloses the maximum latency between every controller, to all other controllers. Similarly, this latency was examined by some placement studies, and it is one of the optimisation foci of this research. Also, the maximum, average and a minimum value of this latency is available under its bar-chart.

C) The Connection Control-Path Latency Versus the Connection Flow-Setup Latency:

The connection control-path and the connection flow-setup latencies are new metrics proposed by this research. These latencies are formulated from the direct analyses of the experimental tests on SDN hardware and emulated equipment, which are explained in Section 3.3.5. ***The connection flow-setup latency*** is defined as the time that required to create connection and pass the first packet between two hosts. This latency includes both latencies of the control and the data paths. While, the ***connection control-path latency*** consists only of the latencies of the control paths for the control packets, which are required to install the flow-rules on the switches.

These two latencies are calculated for every node to all the other nodes of SD-WAN. However, the COVN simulator displays only comparisons between them from every node to the nearest node and the farthest node.

The bottom half of this tab exhibits two stem plots, which introduces these two comparisons. Each plot compares the connection control-path and the connection flow-setup latencies. While, the ***left plot*** introduces a comparison of these two latencies for every node to its nearest node (neighbour). The ***right plot*** reveals the matching between these two latencies from each node to its farthest node in the graph. The reason for selecting the comparisons from every node to nearest and farthest nodes is to present the ratio of affecting connection control-path latency on the connection flow-setup latency for the minimum and maximum path length. Also, the maximum, average and minimum values of these latencies (from every node to all the other nodes) are presented in the space between them.

5.3.6. Sixth Tab (Reliability)

The results and statistics of the reliability are presented in the sixth tab. The three bar-charts present the different parameters that indicate the reliability of the SD-WAN such as the switch reliability, the controller reliability and the control-path reliability, as shown in the Figure 5.12.

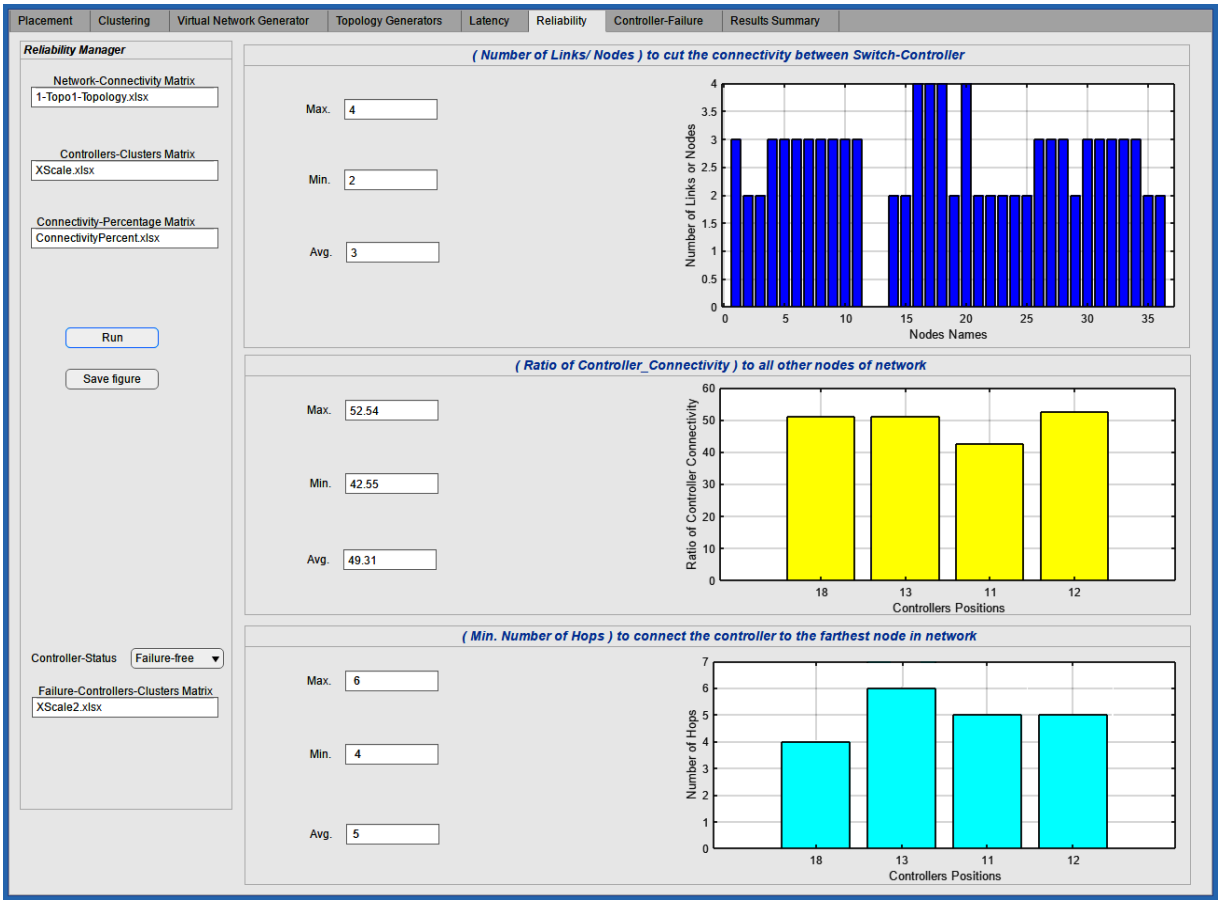


Figure 5.12: The sixth tab of COVN simulator (Reliability).

The maximum, average and minimum values of each parameter are displayed on the left side of the bar-chart in this figure. The required matrices to calculate these parameters are topology connectivity matrix and controllers-clusters matrix which is produced by the COVN algorithm. Also, this tab needs to determine if the reliability computations are failure-free or have a failure status. The reliability parameters are defined in the following:

A) Number of Links/ Nodes to cut the connectivity between Switch-Controller: It is an important parameter used to evaluate the SD-WAN reliability. The higher value of cut-link or cut-node provides more stability for network against switch or link failure because it means there are more paths from the switch to the controller. This parameter is used to refer to the reliability of every switch in the network. The first bar-char at the top of the Figure 5.12 displays the values between every node (switch) and its controller.

B) Ratio of Controller-Connectivity to all other nodes of the network: This parameter indicates, the extent to which the controller location is connected to the network (all nodes). The controller connectivity ratio is extracted from the *Normalised Connectivity* which is calculated by the Equation 4.7 in Section "4.7.3. The Design of COVN Placement Algorithm- Part 1". The middle plot in Figure 5.12 discloses this ratio for every controller.

C) Min. Number of Hops to connect the controller to the farthest node in the network: The minimum number of hops of every path between the switch and the controller represents the reliability of this control-path. The smaller (shorter) control-path refers to a lower probability of link/node failures. The bottom drawing in Figure 5.12 offers the highest minimum number of hops for every controller to its nodes.

5.3.7. Seventh Tab (Controller-Failure)

The seventh frontage is responsible for producing the controller-failure status. It is programmed to simulate the work of COVN placement algorithm when the controller failure happens. It implements that by disabling the active controller/s which is/are given in the input text field named failed controller/s. After that, it uses the next controllers in the controller list that is produced by COVN placement. Then, it reassigns the halted clusters to the nearest backup controller/s. The output of this process is represented by a new controller-clusters matrix that could be used in the latency and reliability tabs to calculate the statistics in the failure status. This tab draws a graph plot which exposes the new controller placement after the controller-failure as exhibited in Figure 5.13 below.

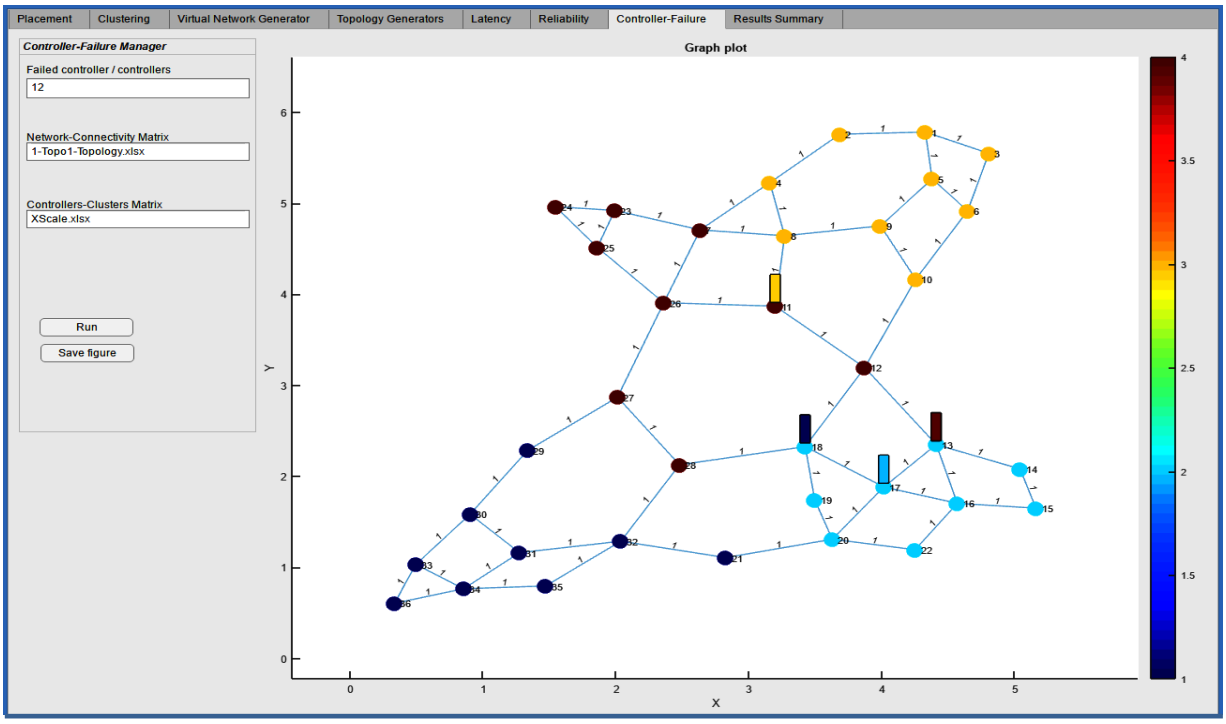


Figure 5.13: The seventh tab of COVN simulator (Controller-Failure).

5.3.8. Eighth Tab (Results Summary)

The last frontage of this GUI creates a table to summarise the maximum, the average and the minimum values of all the latency and reliability parameters that are calculated by this simulator (see Figure 5.14). Furthermore, it has a button to save all the created plots of the simulator in good quality resolution to the specified folder. Also, it has another button to save the files that represent the decision of the COVN placement algorithm such as:

- Controller list file (locations and priority of physical controllers).
- Controllers-clusters for every virtual network in failure-free and failure status (assigning virtual controller for every cluster).

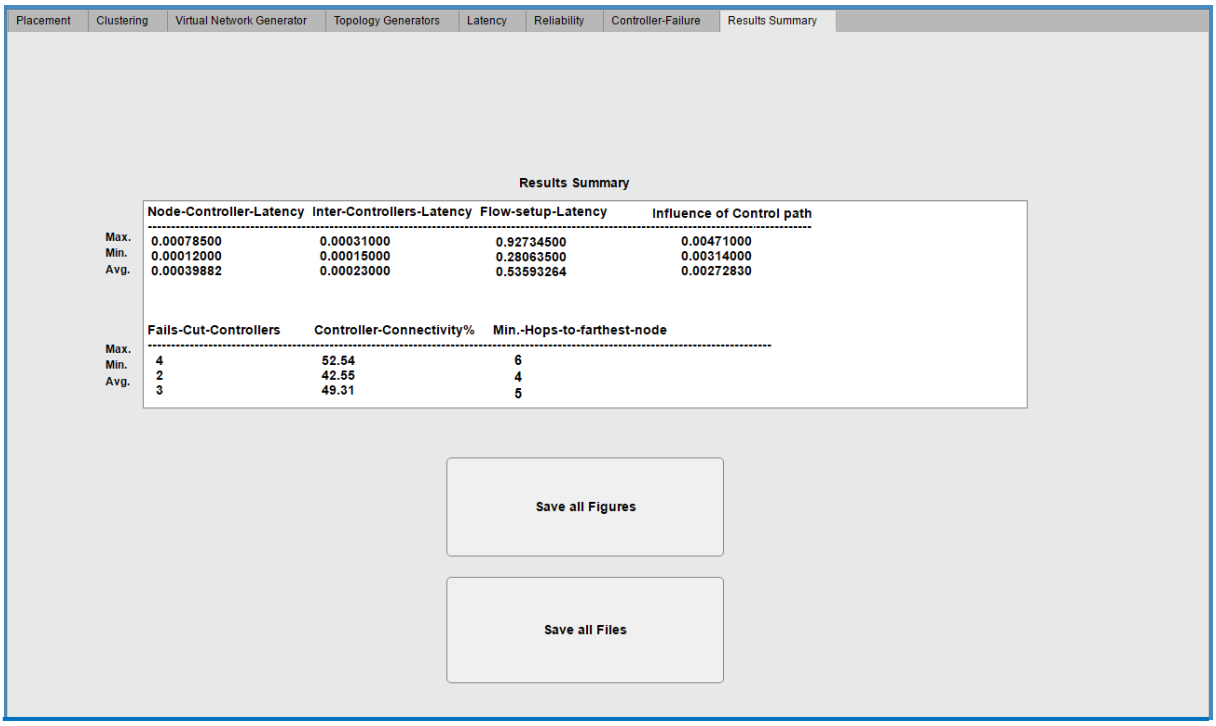


Figure 5.14: The seventh tab of COVN simulator (Results Summary).

5.4. The Developed Mininet Emulator

Mininet emulator provides a flexible environment to implement and test SDN networks. In addition, it supports the results realism and test replication (Heller, 2013). The results realism means the emulation results are close to the results which can be obtained from real network equipment (hardware or software). The test replication indicates that the test could be implemented as much as needed with the same components of the test environment. Therefore, this thesis intends to use the Mininet in some experiments to validate the results accuracy of the COVN simulator. However, it does not support the implementation of controller placement because it cannot emulate

the effect of changing the controller position. For that reason, this mechanism is added to Mininet to enable controller placement emulation. This section explains the problem and its proposed solution; the tools used to implement the solution and which mechanism is used to add the effect of changing controller position.

5.4.1. The Mininet Limitation and the Developed Solution in Mininet

Mininet can connect the controller to OpenFlow switches whether it was internal (on the same machine) or external (on different machines). Also, it can connect the switches to several controllers simultaneously, each group of switches is specified to one controller. The port and IP address of a controller is used to identify the controller for the switches, whereas, only the port number of the switch is used to identify the switches to the controller. ***The problem is***, the ports connecting the switch and controller are not network ports, they are passive ports (Listening port), which are used to communicate between applications. That decimates the ability to establish a link between the switch and the controller and inhabit the possibility to make any specification or configuration for this connection. Therefore, it was not possible to reflect the effect of the controller position in the network. Conversely, all other ports used to communicate between switches or between switches and hosts are network ports. Consequently, the network links can be established and configured with any network parameters like bandwidth, latency and packet losses ratio.

That was the only limitation in Mininet according to the implementation of the controller placement algorithm. It was the motivation to find the solution to this problem. The solution is summarised by tracking the communication between the passive ports (listening ports) until the socket between the controller and switch is established, after which the active port is created. The active port is used to identify the connection and apply the traffic control mechanisms on that connection. The tools which are used for tracking the connection and apply traffic control are explained in Section 5.4.2.

5.4.2. The Tracking Communication and Traffic Control Tools

A) Tracking Communication Tools

Mininet provides a wide range of tools for monitoring and debugging the communication of SDN network. It does this by integrating many well-known tools of the real network to support network monitoring realism. Also, it allows the network

developers to perform the monitoring in exactly the same way of the real network. For example, Mininet uses the OVS to emulate the behaviour of OpenFlow switches. Therefore, it can use all the tools which have been developed to debug the OVS switches like **ovs-dpctl** (OpenvSwitch, 2016b), **ovs-appctl** (OpenvSwitch,2016a), **ovs-ofctl** (OpenvSwitch,2016c) and **ovs-vsctl** (OpenvSwitch, 2016d)**.** Also, it uses famous monitoring tools such as Netstat (Durgin *et al.*, 2005), tcpdump (Van *et al.*, 2016), Wireshark (Wireshark, 2016b) and tshark (Wireshark, 2016a). All the mentioned tools were tested to capture the needed information, but the most suitable one was the **tshark**. That is because it can capture the traffic continuously in contrast to the other OVS debugging tools and Netstat monitoring tools which could only discretely capture the status of switches. Also, it was easier than tcpdump in filtering the captured data. Finally, the tshark is used to extract the port number of the switch which is created and activated when the socket between the switch and controller is established.

B) Traffic Control Tools

There are two tools of traffic control used to solve the problem of the limitation of controller placement in Mininet they are: the **Linux Traffic Control** (tc) and **Network Emulation** (netem). Combining these tools allows for applying all traffic impairments on emulated nodes and links (Tim De Backer, 2014). These tools can get a proficient level of traffic realism in complex or distributed emulation of the network.

On the one hand, these tools were already applied to the emulated links between the switches or between the switches and their hosts in the Mininet (Mininet Team, 2016). On the other hand, these tools were not applied on the links between switches and controllers. Since, these links are not considered as network links. Therefore, it was imperative to recall these links (capturing their port number) and apply these tools to them in order to allow their configuration. The **tc** is an authoritative tool for shaping, classifying, scheduling and prioritising the network traffic. That is because, it is basically a based **queuing discipline** (qdisc) algorithms, which has had a long development to optimise the network traffic scheduling. The tc tool can apply any scheduling algorithm and set the transmission rate (bandwidth). However, it is not able to add all traffic deviations such as latency and jitter. Therefore, the Netem tool is used to extend the abilities of tc tool as it allows it to define and set more traffic impairments (Tim De Backer, 2014).

5.4.3. The Applied Traffics Control Mechanism

Changing the controller position in SDN network affects the communication between the controller and its switches. Therefore, a different path between them should be used when the controller position is changed. In a real network, this occurs automatically due to the change of the control path, Mininet emulator does not change it automatically because Mininet emulation mechanism separates the control path from the data path. This means each switch is connected directly to the controller by a separated link (control link), which ignores the controller position in the network (see Figure 5.15).

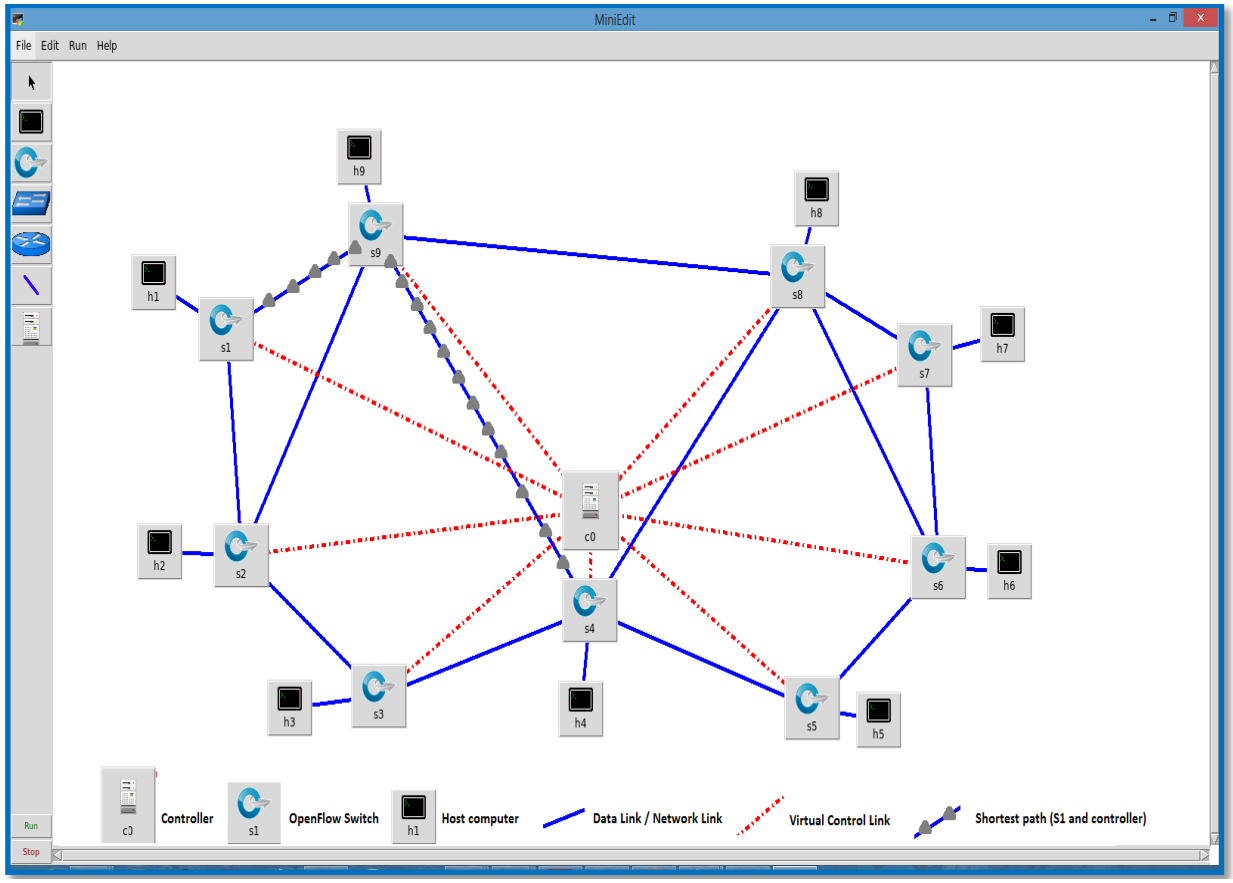


Figure 5.15: SD-WAN in Mininet that demonstrates the difference between the virtual control path and data path.

Therefore, the effect of the controller position should be added to Mininet by the following steps:

1. The path between the controller position and each switch should be identified using a routing algorithm, which computes the optimal route between the two points.
2. Communication properties of each path should be converted to traffic control rules.
3. The traffic control rules of each path should be applied to the coordinate control path.

The communication properties are represented by communication latency, jitter, and losses packet ratio. The latency variation is caused by many parameters such as the links bandwidth, the number of passed hops (switches) and switches' transmission ratio. All these parameters are considered for computing the path properties in the introduced solution.

Figure 5.15 shows an example of SD-WAN which is created by Miniedit, a primitive GUI of Mininet emulator. The example demonstrates the difference between the data path and control path in Mininet. Note that the switches and hosts are connected using blue lines (Data links/Network links), while all switches are connected directly to the controller using red lines (Virtual Control Links). Therefore, the controller position does not affect the control path. The example assumes that the controller is located in the position of switch four (S4). For instance, if the controller wants to communicate with the switch one (s1) using shortest path. It should pass the path of two links through switch two (s2) (line with grey triangles), while it communicates now directly through the red link. That is not the real behaviour of SDN network and does not reflect the effect of controller position. Therefore, the properties of the blue path are calculated, converted to traffic rules and applied on the red path to achieve more realism and emulate the effect of controller position.

5.4.4. The Programmes of Implementing the Mininet Developments

This section shows the main programmes that are used to implement the above-mentioned procedure.

1. Executing Matlab programme (Latency.m) to calculate the latency of the shortest path between every node and its controller. This programme will create switch-controller latency matrix to be used to configure the control link in Mininet using tc tool.
2. Running python programme (portcap.py) on the Mininet-Virtual Machine before starting the Mininet topology to capture the switches names and ports.

```
sudo python ./ portcap.py
```

The output is (outfile.txt), which contains the ports number that connects the switch to its controller for all the switches of topology.

3. Starting the Mininet topology.

```
sudo mn --custom ~/mininet/custom/Topo1-2.py --topo mytopo  --mac  --controller=remote,
  ip=192.168.0.44, port=6633 --switch ovs, protocols=OpenFlow13
```

4. Running python programme (applyTC.py) to run the tc on Mininet-Virtual Machine.

```
sudo python  ./ applyTC.py
```

5. Implementing the required tests as will be explained in the results in the chapter five.

5.5. The Assisting Programmes

There are two additional Matlab programmes that perform the initial preparation to simulate or emulate the real world networks which are obtained from the ***Internet Topology Zoo*** (Knight *et al.*, 2011). The Internet Topology Zoo is a network database which is created from the real network data that is made available by the operators. These two programs are explained as below:

A) Distance Extracting Program:

This programme reads the position of the node from the GML file which is downloaded from the internet topology zoo and calculates the distances between these nodes using the Vincenty equations. This formula is not very accurate for computing the distance on the earth's surface because it treats the Earth as a sphere rather than a spheroid which has two radii a and b, but it could be treated as the most accurate formula (Veness, 2016). This programme creates a distance matrix for the selected topology in order to use it in the COVN simulator or Mininet emulator as well.

B) Mininet Code Generating Programme:

Another Matlab program is used to read the real world topology from the GML file and generate a python code which builds a topology in Mininet emulator. This programme does not only produce a Mininet topology faster and easier, but also avoids the error that could occur through the producing process.

5.6. The Implementation Environment

The computers which are used to implement the COVN simulator and Mininet emulator have the following specifications: (1) corei7 - 2.4GH processor; (2) 16GB memory; (3) 125 GB-SSD hard drive; and (4) Windows 8.1, 64-bit operating system.

The emulator used the HP VAN SDN controllers. The emulator also used the OpenFlow protocol version 1.3 (OF 1.3) because it has the features, which are required to implement the COVN placement algorithm. OF 1.3 also is the dominated protocol for the latest version of SDN controllers and switches.

5.7. Summary and Conclusion

This chapter presents the necessity for using the created COVN simulator to compute the placement decision and to calculate the resultant latency and reliability parameters. Then, it emphasises the importance of applying some tests on the modified Mininet emulator to improve the validity of the simulator results. This chapter also explains in detail about every one of the eight tabs of the COVN simulator including its use, its way of working, its output and functionality. In addition, the chapter illustrates the functionality shortage of Mininet, which hinders the implementation of the controller placement and how this study overcomes this. Furthermore, the two assisting Matlab programmes that prepare the topology distance matrix and topology Mininet code are provided here. Finally, the specifications of the computers used are mentioned at the end of the chapter.

CHAPTER SIX
RESULTS AND EVALUATION OF COVN PLACEMENT

6.1. Introduction

The results of implementing the COVN placement algorithm are demonstrated in this chapter. The simulator results are explained in detail, and then results are evaluated in two different ways. The first evaluation is formed by implementing the placement decision, which is generated by the COVN placement over Mininet emulator to validate the latencies which are calculated by the COVN simulator. The second evaluation, by comparing the latency and the reliability parameters that are produced by the COVN placement algorithm with those which are generated depending on the POCO tools using the statistics of COVN simulator. The chapter starts by showing the used topology and the reasons for choosing them. It then explains the tests of the COVN simulator by clarifying: their objectives; their execution strategies; and their results. Further to that, the Mininet results are introduced. The comparison between COVN placement and POCO placement algorithms is then discussed. Finally, the chapter is summarised and concluded.

6.2. SD-WAN test Topologies and Parameters

The COVN placement is applied over six SD-WAN topologies that have the different sizes (different number of nodes) and geographical expansions (different length of links). The reasons for that are: examining the performance parameters of the COVN placement algorithm on different load (number of nodes), such as load balancing and computation time; Also, exploring the limits of latency optimisation in relation to the length of links.

Two of these topologies are generated by the topology generator, while the other topologies are real-world networks, which are downloaded from the internet zoo topology database (Knight *et al.*, 2011). Figure 6.1 shows these topologies with coloured nodes that indicate their closeness to all other SD-WAN nodes. The first topology named Generated-1 and it is produced by the COVN simulator with a maximum

length of links equal to 20 Km, which is the optimal length of the link for the COVN placement algorithm. Conversely, the second topology Internet2 expands over very wide distances (from 42 to 1612 Km) which helps to find the COVN algorithm limitation in term of the maximum length of the link. Furthermore, the other topologies, which are India, US carrier, Generated-2 and Kentucky, have incremented a number of nodes to determine to which extent the COVN algorithm could keep providing a quick and probable performance. Most networks in the zoo topology database have a small size, but only a few topologies have big sizes. These topologies are selected to be used, however the size of large topologies jumps from 158 nodes (US carrier) to 754 nodes (Kentucky) therefore, the topology called Generated-2 is created with 400 nodes to cover this jump (see Table 6.1).

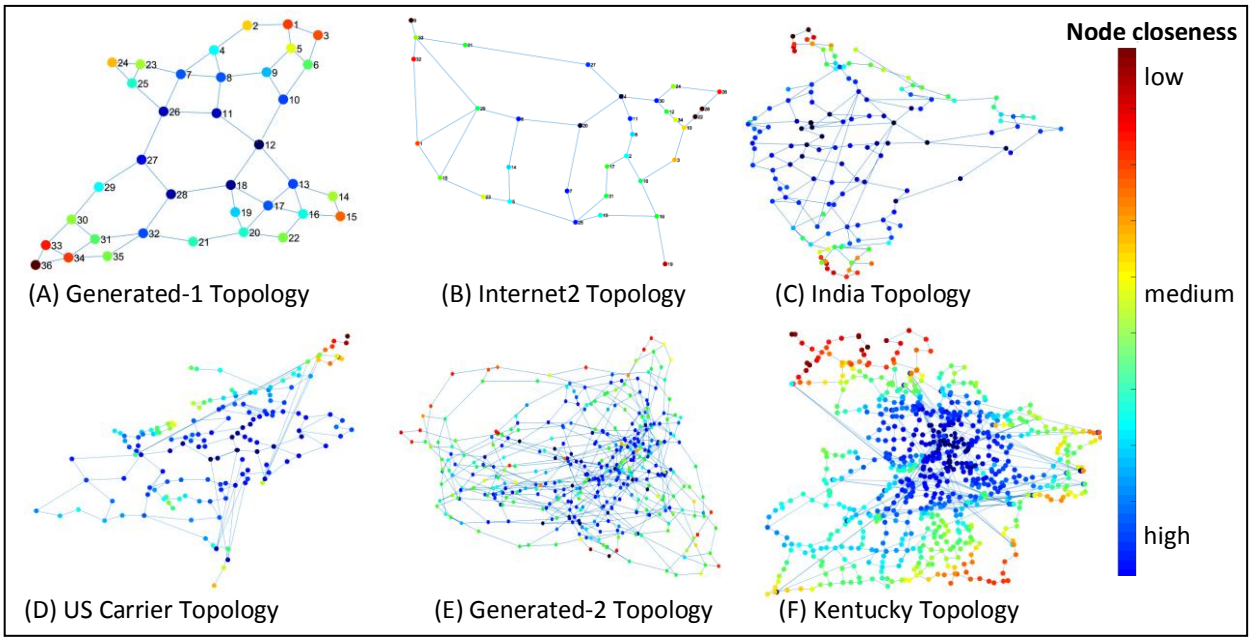


Figure 6.1: The six topologies used for testing the COVN placement algorithm.

Topology Name	Number of Nodes (Switches)	Minimum length of link (Km)	Maximum length of link (Km)	Number of Links
Generated-1	36	7	20	54
Internet2	34	42.920	1612.096	42 *40 links >100 Km*
India	145	1	152.5	187 *110 links >100 Km*
US Carrier	158	3.870	833.7455	189 *Only 47 links >100 Km*
Generated-2	400	6	20	559
Kentucky	754	1	909.801	895 *Only 103 links >100 Km*

Table 6.1: The specifications of the six topologies used for testing the COVN placement algorithm.

In addition, Table 6.2 presents the test parameters that could be changed through the conducted tests.

NO.	Parameter	Explanation
1	**Topology of Physical Network**.	The tests are applied to several SD-WANs.
2	**Weight of Closeness-Reliability**.	The total weight equals 1, which should assign a specific ratio to the closeness and the reliability.
3	• **Number of Physical Controller**. • **Load**: ✓ Number of switches. ✓ Request per second of every switch.	When placing the physical controllers, the number of active and backup controllers or the maximum expected load for the entire network (all VNs) should be determined.
4	• **Number of Clusters**. • **Load per Cluster**: ✓ Number of switches. ✓ Request per second of every switch.	When placing the VN, the number of clusters or the maximum load of the cluster for every VN should be specified.
5	**Topology of VN.**	For every SD-WAN topology several VNs are applied, then the connectivity matrix of every applied VN should be provided.

Table 6.2: The parameters that could be changed when implementing the COVN placement algorithm.

6.3. The Results of the COVN Simulator

This section describes the test objectives, test strategy and test results for the conducted tests that are examining the COVN placement algorithm. There is six categories of the conducted test as follows:

6.3.1. The Metrics of COVN simulator

This is the first category of tests, which intends to present and explain the basic metrics of COVN simulator in order to validate some of the assumptions that are considered when the COVN algorithm is designed. These assumptions are:

- The Node-Controller latency is relatively small;
- The algorithm selects near positions for hosting the physical controllers to improve the latency of inter-controller communication;
- The connection control latency has a small portion in connection flow-setup latency;
- The nodes which are selected by COVN algorithm for hosting the controllers have the best reliability parameters to other nodes of SD-WAN.

- Finding the backup controller is performed without recalculating the controller placement or clustering the topology (no additional computational complexity).

These objectives are demonstrated by presenting the complete steps for implementing COVN placement algorithm on single topology (Generated-1) in failure-free and failure cases. These steps produce the figures and statistics of latency and reliability which should be concluded to validate the above assumptions. The steps for implementing COVN placement algorithm are as follows:

A) Applying the placement of physical controller produces their number and their positions, as shown in Figure 6.2. In this topology, there are five physical controllers, and their locations are ordered in the following sequence 12,18, 28, 11 and 26 (white rectangles). The first four controllers are the active controllers, while the last one represents the backup physical controller. The node's colour shows its closeness to all the other nodes of the graph.

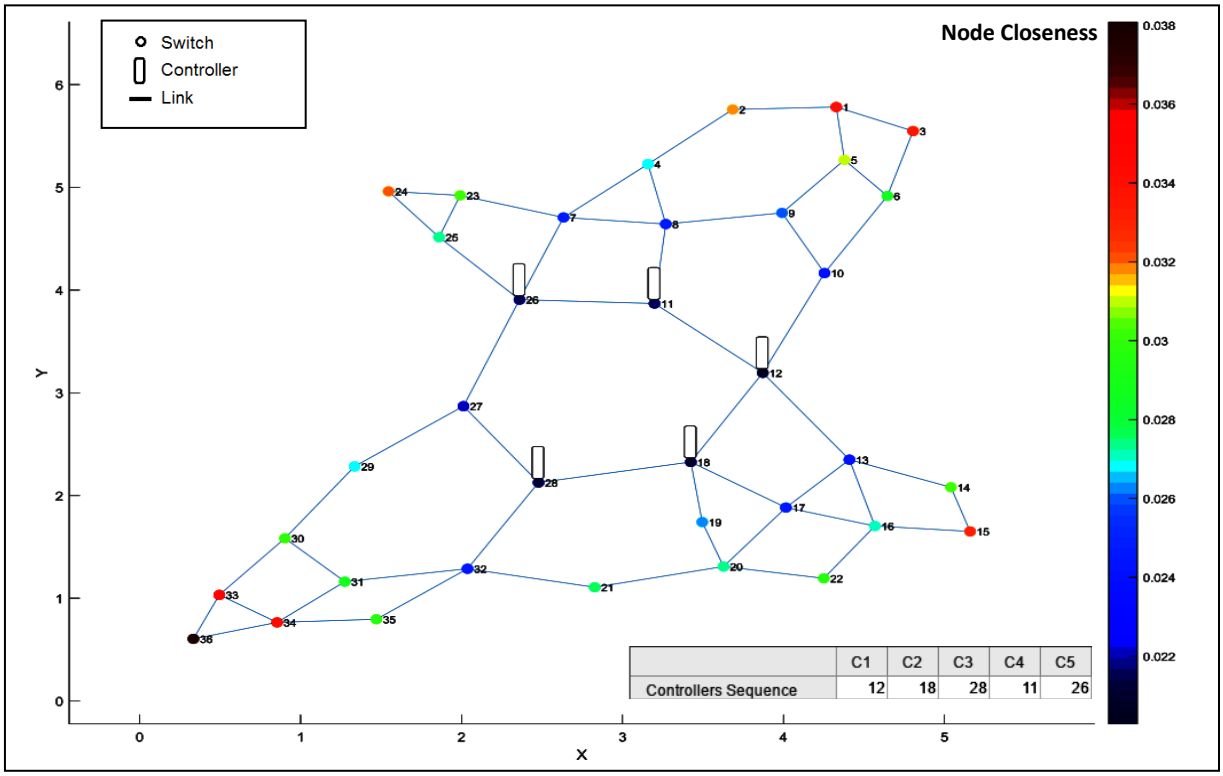


Figure 6.2: The placement of physical controllers for Generated-1 topology.

B) Performing the placement of the virtual controller, clustering the virtual network and assigning a virtual controller for every cluster. Figure 6.3 Shows that the Generated-1 topology has a single VN and this includes the complete physical topology; it is then partitioned into four clusters, each of them has nine nodes (switches). ***This discloses the ability of COVN algorithm to balance the controller's load*** (more detail will be

presented in Section 6.3.3). The nodes of every cluster have the same colour as its controller, and the value of colour-bar indicates the number of clusters.

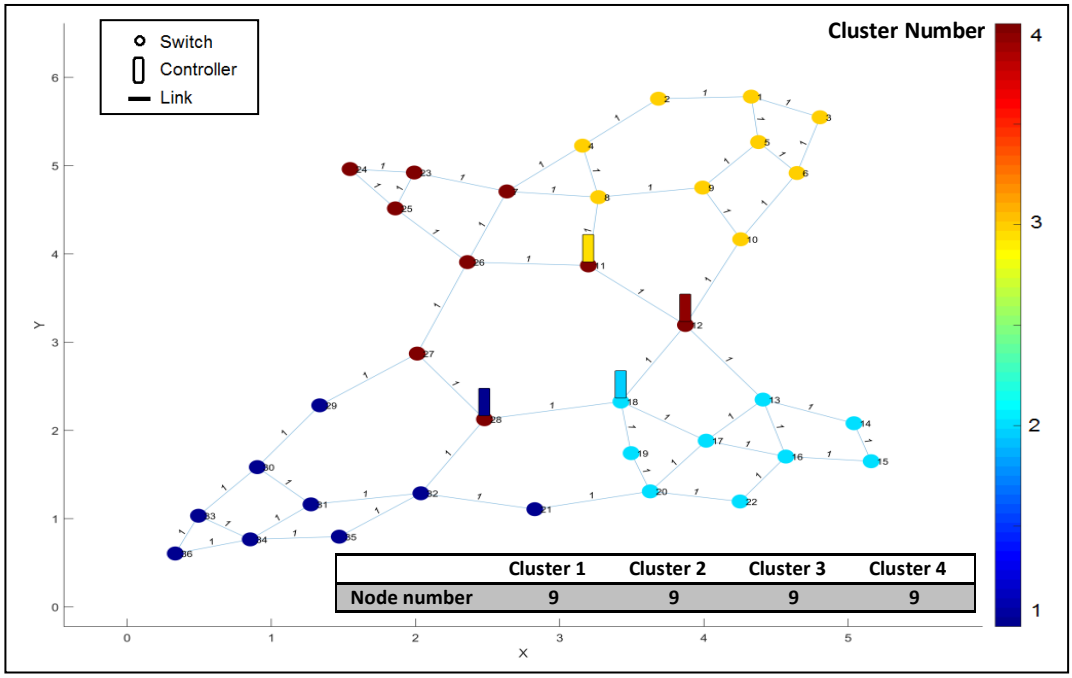


Figure 6.3: The placement of virtual controllers for Generated-1 topology.

C) Executing the latency statistics to calculate four metrics as follows:

(C-1) Node-Controller Latency Metric: It is a fundamental metric for evaluating the controller placement because the switch-control path changes in the consonance with the controller's positions.

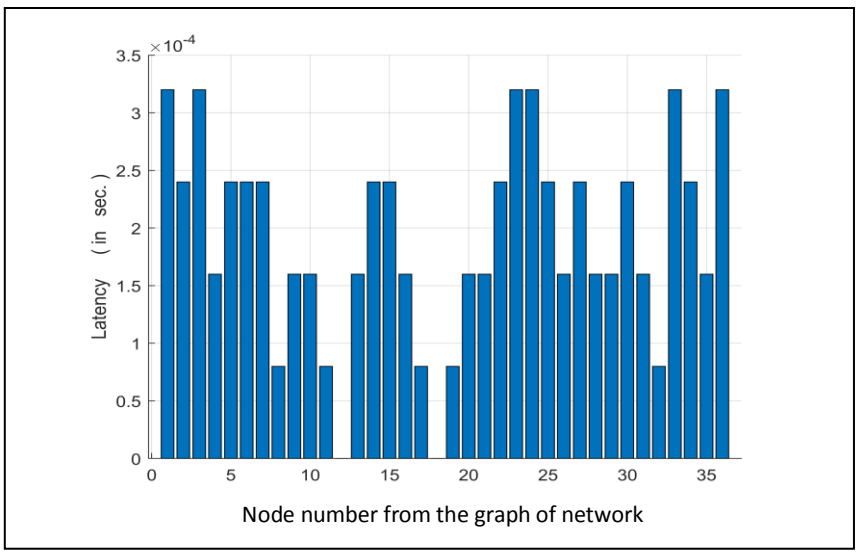


Figure 6.4: The node-controller latency for all switches of Generated-1 topology.

Figure 6.4 presents the latency from every switch to its controllers, for all switches of the network (36 switches). This figure shows that the latency could vary from (***80.005 μs***) at nodes 1, 3, 23, 24, 33 and 36 to (***320.02 μs***) at nodes 8, 11, 17, 19 and 32. This

reflects that the latency of the close switches is a quarter of the farthest switches latency. ***However, the values variation is relatively small because it is in 'microsecond' while the flow-setup latency is almost in 'millisecond'***.

(C-2) Maximum Inter-Controller Latency Metric: Is the second basic metric, which is significantly improved to the optimal limit and avoids degrading the other metrics. That is because its nodes not only reduce the latency of the control path but it also optimises the usage of network resources (links and switches) when converging the controllers' rules and databases. Figure 6.5 shows that the Maximum inter-controller latencies are altered from (***80.005 µs***) at controller's positions 12 and 18 to ***(240.015 µs)*** at controller's positions 11 and 28. The difference in these latencies is not big because the nearest controller's positions can reach all other controllers by the maximum path of one hop, while the farthest controller's positions communicate with the others with three hops. ***The COVN algorithm intents to keep the inter-controller path with a minimum number of hops.***

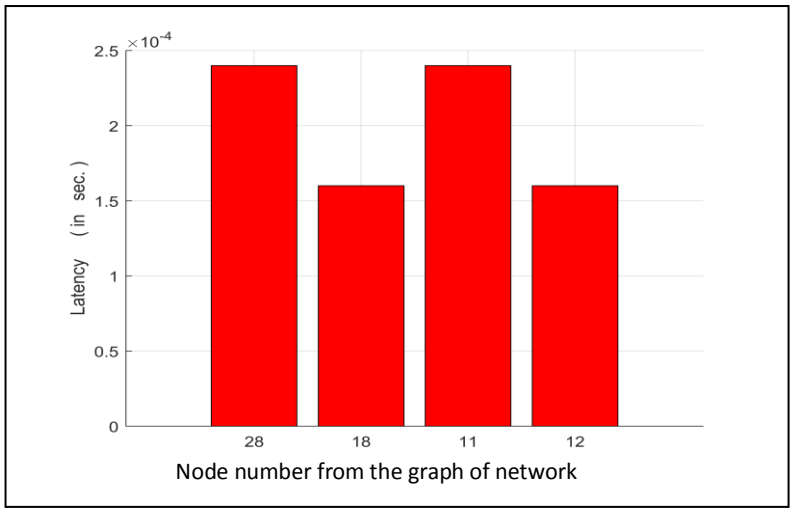


Figure 6.5: The maximum inter-controller latency for every controller to all other controllers of Generated-1 topology.

(C-3) Connection Control-Path Latency Metric Versus Connection Flow-Setup Latency Metric: The connection control-path latency includes all latencies that are required to install the forwarding rules to all the switches on the path of the flow, to deliver the first packet between two hosts. The connection flow-setup latency represents the complete time that is consumed to send the packet from the sources host to the destination host, which consists of the controller latency, the connection control-path latency and the connection data-path latency (see Chapter five, Section "5.3.5. Fifth Tab (Latency)"). These two metrics are plotted together as a stem-chart with the aim to show the ratio of

contributing the first one on the later. These two metrics are calculated for the communication of every node to all other nodes. However, the presented stem-charts show only the comparisons of these two latencies from every node to the nearest and farthest nodes in order to demonstrate the minimum and maximum ratio of the effect (see Figures 6.6 and 6.7).

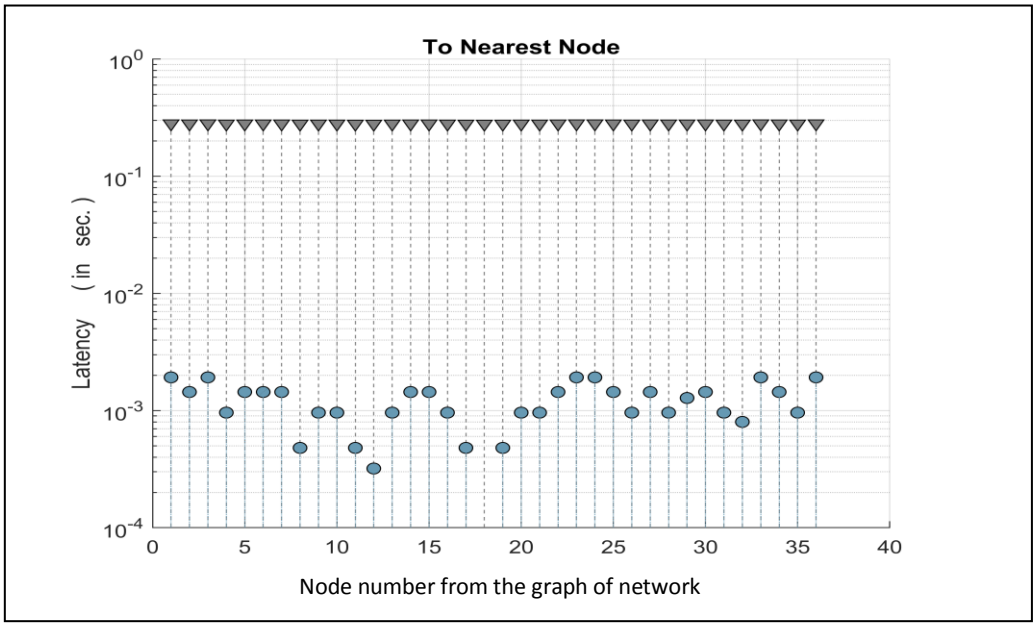


Figure 6.6: Connection control-path latency metric versus connection flow-setup latency metric to nearest node of Generated-1 topology.

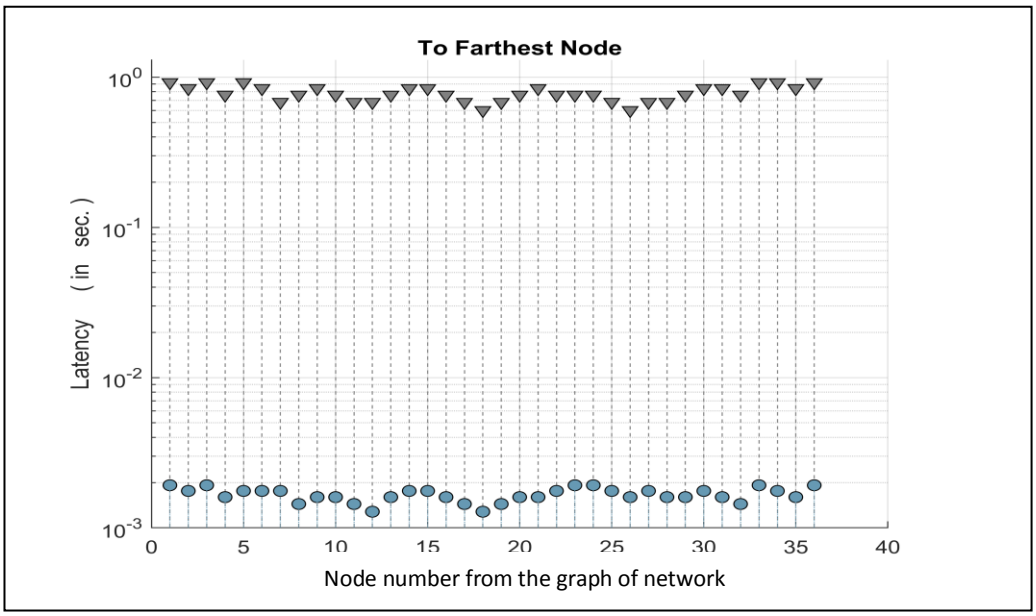


Figure 6.7: Connection control-path latency metric versus connection flow-setup latency metric to the farthest node of Generated-1 topology.

Both Figures above reveal that the connection control-path latency is much smaller than the connection flow-setup latency. One clean example is shown in Figure 6.6 which compares the maximum values of connection control-path latency (***1.920 ms***) and the

connection flow-setup latency (***923.600 ms***), ***which validate that the effect of the first latency is small upon the second one.***

Also, Figure 6.6 realises that the connection flow-setup latencies are very close to each other (between ***280.4 and 282.01 ms***) because the variation of the connection control-path latency is tiny (between ***0.32 and 1.9 ms***) and the variation of the data-path to reach the nearest node is small (one link). While, in Figure 6.7, the connection flow-setup latencies have a larger variance (between ***602.3 and 923.6 ms***) due to the bit higher variance in the values of the connection control-path latency (between ***1.3 and 1.92 ms***) and the variation of the data-path to reach the farthest node is higher (from 5 to 6 hops). ***The comparisons of the previous ranges find that the ratio of effecting the connection control-path latency is always small on the connection flow-setup latency.***

Finally, it is noteworthy that the average values of these two metrics are provided by the simulator, however they are not plotted.

D) Executing the Reliability statistics calculate three metrics as follows:

(D-1) The *Metric for the Number of Links/Nodes to Cut the Node-Controller Connection*: The first metric defines the ***reliability of the switches***, which is the number of links or nodes that cut the path between the switch and its controller.

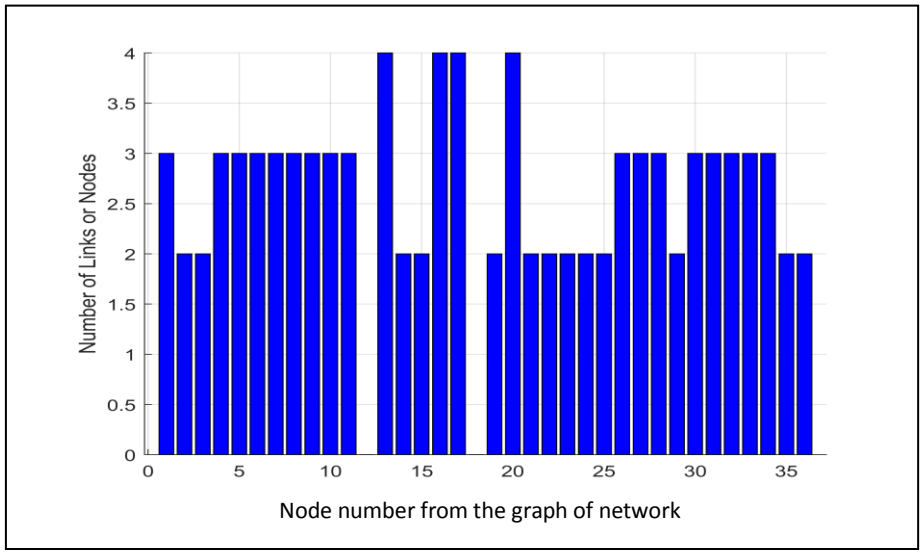


Figure 6.8: The metric for the number of links/nodes to cut the node-controller connection of Generated-1 topology.

Figure 6.8 presents the metric which demonstrates that all nodes (switches) are connected to their controller by at least two different paths. Therefore, the connection between them is only halted, if two nodes or links fail in these two different paths at the

same time. ***This proves that all switches have acceptable reliability against control-path failures when applying the COVN placement algorithm***.

(D-2) The *Ratio of the Controller Connectivity Metric*: This is the second metric that precisely identifies the ***reliability of the controller locations***, which is the ratio of the connectivity of controller to all the other nodes of the network (see Chapter five, Section "5.3.6 Sixth Tab (Reliability)").

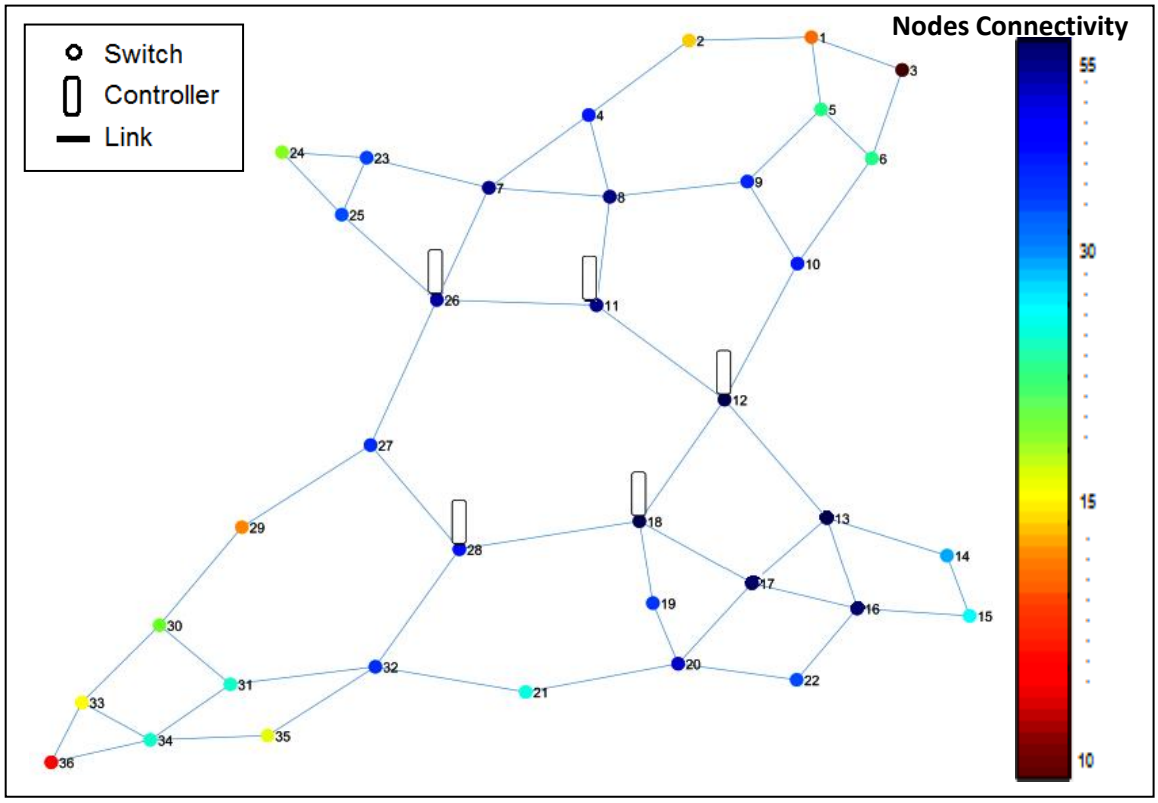


Figure 6.9: The ratio of the nodes connectivity for Generated-1 topology.

The colours of the nodes in Figure 6.9 indicates the ratio of the node connectivity. These values are varying from the smallest value at node 3 (9.864) to the highest value at the node 17 (53.353). This range differs for every topology according to the connectivity of its nodes (number of links). Calculating this range for a specific topology helps to identify the nodes which have the acceptable reliability and could then be used to host the physical controllers. As mentioned before, the COVN algorithm computes the physical controller placement according to closeness against the connectivity by specifying the desired weight for each of them. The strategy of choosing the optimal weights will be discussed in the next Section 6.3.2. It is noteworthy that the controller

positions which are selected for the Generated-1 topology have an acceptable ratio of controller-connectivity, as shown in Figure 6.10.

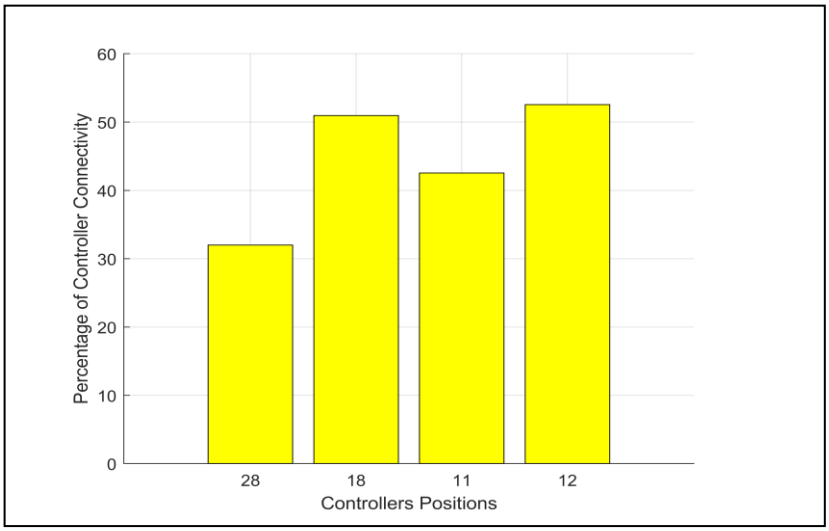


Figure 6.10: The ratio of the controller connectivity metric of Generated-1 topology.

For more detail, Figure 6.10 reveals that the highest connectivity ratio of controller's nodes is 52.539 at node 12, while the lowest one is 32.003 at node 28. These values of connectivity ratio reflect good reliability for controller locations. ***This proves that the COVN placement algorithm produces an almost reliable placement. Bearing in mind that, increasing the weight of the reliability will increase the reliability of the select controller's nodes.***

(D-3) The Metric for the Minimum number of Hops to the Farthest Node: This metric indicates ***the reliability of the control-path*** between the switch and its controller. Figure 6.11 shows that the minimum number of hops from every controller to the farthest node (called ***node eccentricity*** in graph theory (Rosen, 2012) varies between four and five hops. While, the minimum number of hops between the farthest two nodes in the network (called graph diameter) is ten hops. ***This shows that control-path is almost equal for all controllers. Also, it is near to the half value of the network diameter which proves that the control-path is much shorter than the diameter. The shorter control-path reduces the probability of node/link failure that could occur in it.*** Also, keeping the length of the control-path stable is important for VN in term of reliability and latency. That's because VN frequently changes its topology and the present controller placement constrains the control-path within four to five hops. In addition, this placement facilitates the reliability of assigning the backup controller because it is placed as much as possible near to the active one.

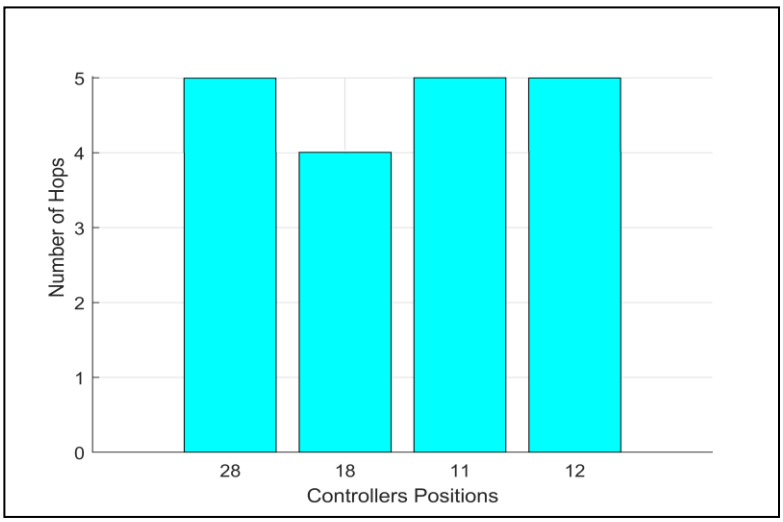


Figure 6.11: The metric for the minimum number of hops to the farthest node of Generated-1 topology.

D) Applying the controller-failure needs to identify the fail controller, then the algorithm picks the next controller from the controllers' list and assigns it to the required cluster. This operation is fast and efficient because it does not cost additional computation and the backup controller has similar characteristics to the original one in term of latency and reliability parameters

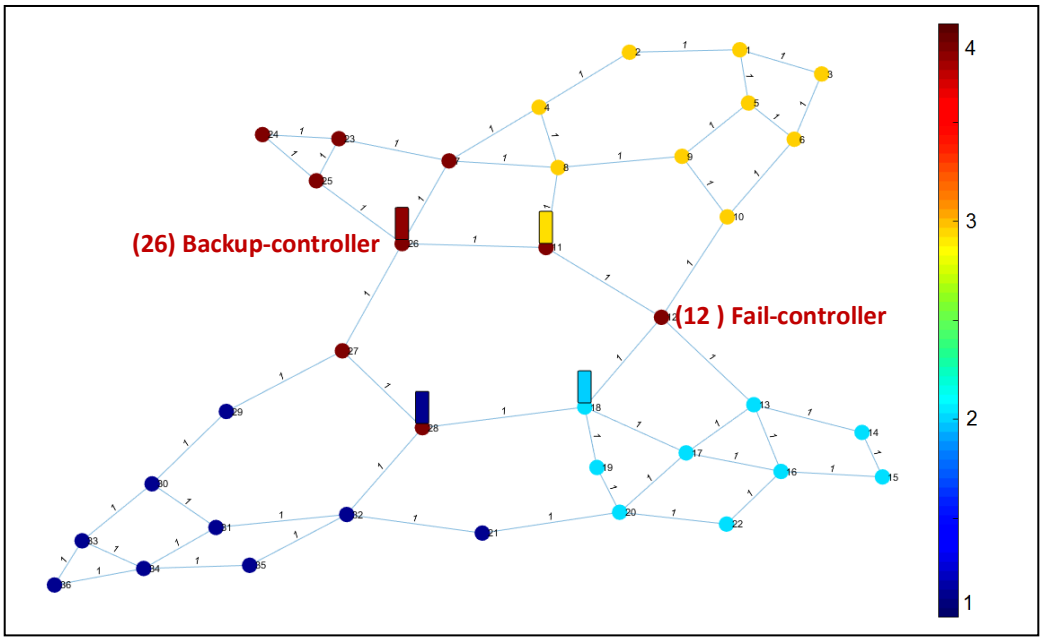


Figure 6.12: The implementation of controller-failure on Generated-1 topology.

Figure 6.12 ***shows that the control of the brown cluster (fourth cluster) is changed from fail-controller at node 12 to the backup-controller at node 26 without reconstructing or modifying the clusters of SD-WAN.*** Also, the figure shows that the backup controller is still close to the cluster nodes and other controllers as well. The detailed comparisons of latency and reliability in the failure-free and failure cases will be

discussed in Section 6.3.4. While, this section focuses on showing that the method of the recovering the controller- failure does not require any recalculating for the placement or the clustering of SD-WAN.

At the end of Section 6.3.1, It is noteworthy that, this chapter will use only the maximum, average and minimum values of the above metrics to represent them in next comparisons instead of using the full metrics for clearer presentation and conclusion. These three values could express these metrics fairly and make it easier to track their changes through the comparisons.

6.3.2. The weight of Closeness against Reliability

The second category of tests is conducted to explore a method for determining the optimal weights of the closeness and the reliability. The test strategy is summarised as follows:

- Using four active physical controllers for all tests.
- Considering the complete topology as a single VN, which is partitioned into four clusters for all tests.
- The COVN placement is applied to the network with five different weights of closeness and reliability, as shown in Table 6.3.

Closeness weight	Reliability weight
1.0	0.0
0.3	0.7
0.5	0.5
0.7	0.3
0.0	1.0

Table 6.3: The weights of closeness and reliability for the second tests category.

- The maximum, average and minimum values of the latency and reliability metrics are compared for the five tests.
- Determining the weights of the test that produces the best metrics as the optimal weights.
- Implementing the same steps over three networks (Generated-1, Internet2 and India), to notice how these weights could differ for every network.

A) The weights of the Generated-1 topology: Applying the previous strategy over this network, produces the following results, in Figures 6.13 and 6.14 below:

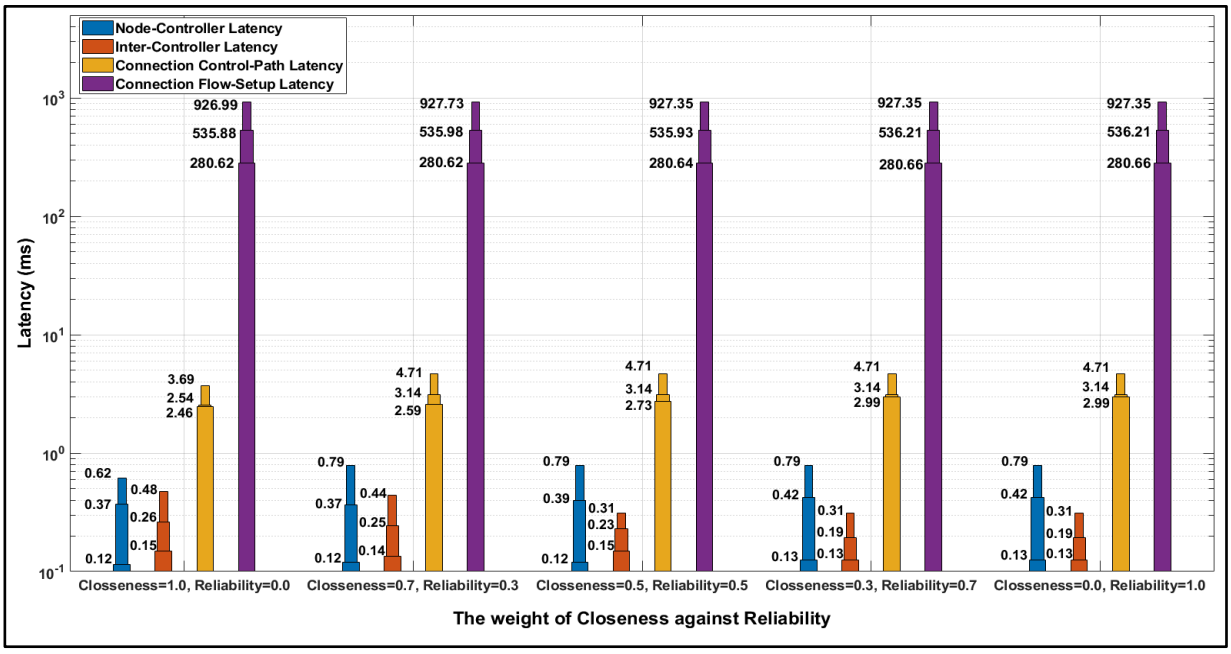


Figure 6.13: The comparison of the minimum, average and maximum values of the resulted latency metrics for the closeness-reliability tests of Generated-1 topology.

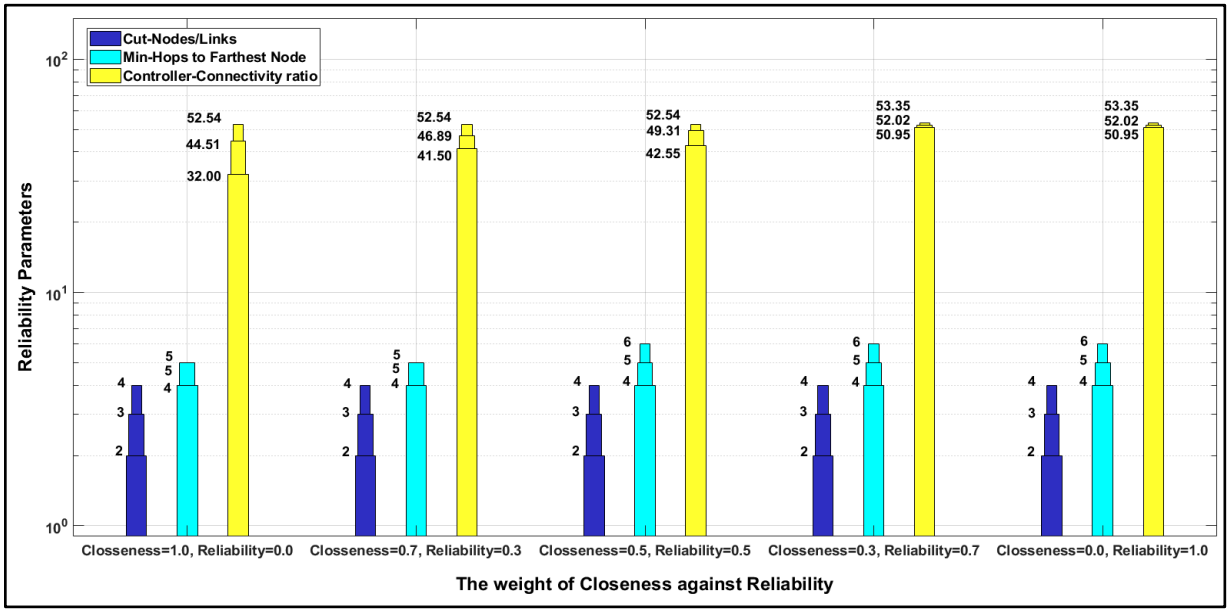


Figure 6.14: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the closeness-reliability tests of Generated-1 topology.

1) Describe the bar-chart of latency metrics:

Figure 6.13 introduces the minimum, average and maximum values of one latency metric in each bar of the plot. The bar-chart includes five groups. Every group represents the four-latency metrics of a single test with specific weights of closeness and reliability (see Table 6.3). The four metrics of latency are the node-controller latency; inter-controller latency; connection control-path latency; and connection flow-setup latency.

2) Describe the bar-chart of reliability metrics:

Figure 6.14 offers the minimum, average and maximum values of one reliability metric in each bar of the chart. The five groups of bars interpret the three-reliability metrics, which are: the cut-nodes/links; minimum-hops to the farthest node; and the controller-connectivity ratio (see Section "(D) Executing the Reliability statistics" above). Every group of bars represents the reliability metrics of a single test with specific weights of closeness and reliability (see Table 6.3).

3) Discussion of the latency metrics results:

For this topology, the latencies are smaller when the closeness weight is higher, except the inter-controller latency, which is smaller when the closeness weight is lower. Also, it is clear in the Figure 6.13 that the maximum value of the node-controller and the connection control-path latencies stays fixed after the second test, while its average and minimum values have a tiny increment. Similarly, the worst-case connection flow-setup latency (maximum value), stops increasing after the second test, while the average and minimum values are raised in microseconds. ***This indicates that it is possible to choose any of the last four closeness ratios to have a similar effect on the latency metrics. Even though the first closeness ratio (1.0) produces the best latency metrics, it is not preferred one to be used because the reliability ratio equals zero.***

4) Discussion of the reliability metrics results:

In general, the reliability metrics raises with increasing the reliability weight. Figure 6.14 demonstrates that the maximum value of the minimum-hops to the farthest node (second bar) increases after the second test (reliability=0.3). Also, it exposes that the values of the controller-connectivity ratio are not far from each other along the five tests. This implies that the only noticeable decline in the reliability parameters appears in test three where the maximum length of control-path (the minimum-hops to the farthest node) becomes longer (from 5 to 6 hops).

5) Defining the optimal weights of Closeness and reliability:

According to the discussion of the point 3, the weights of the second test (closeness=0.3, reliability=0.7) produces the second optimal latency, which does not increase even by increasing the closeness weight. Also, from point 4, the increment of the controller-connectivity ratio is relatively small along all tests, while after the second test the

maximum length of the control-path is increased. Therefore, the weights of the second test are considered the optimal weights to be used in the COVN placement algorithm.

B) The weights of the Internet2 topology: Implementing the similar steps on the Internet2 OS3E topology, generates the results which are presented in Figures 6.15 and 6.16 below (see the description of the figures in the Section "6.3.2, (A), points 1 and 2"):

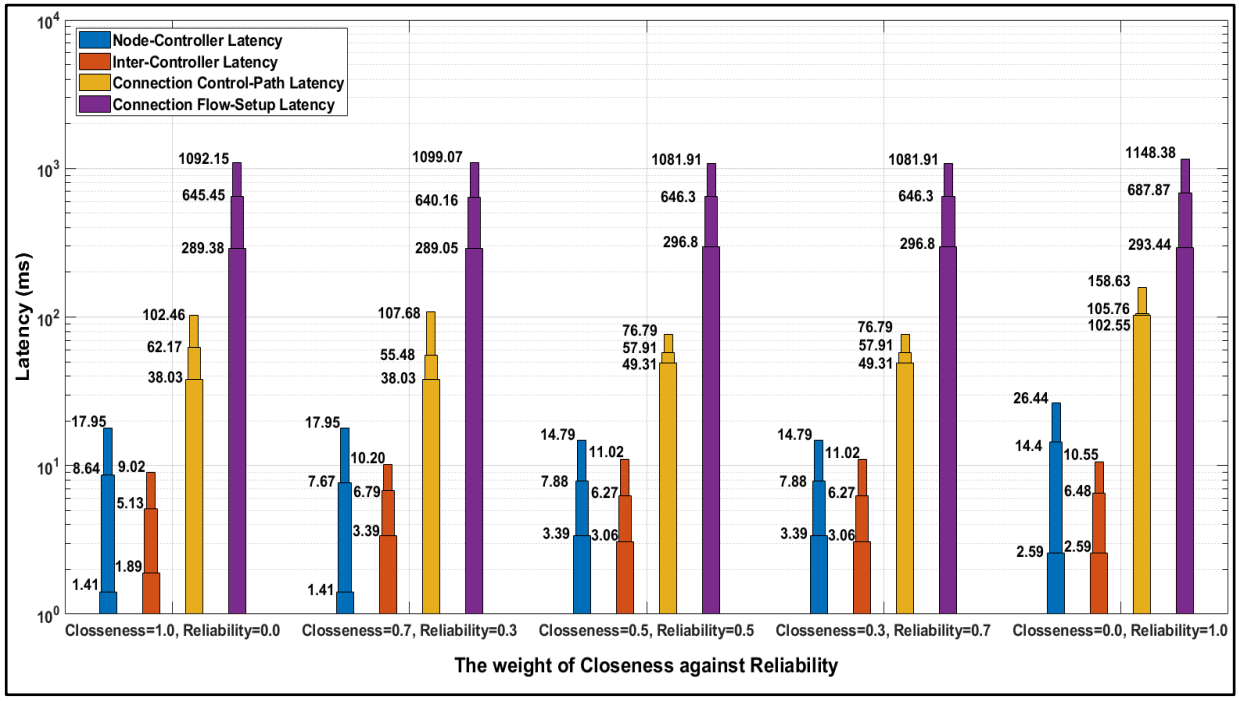


Figure 6.15: The comparison of the minimum, average and maximum values of the resulted latency metrics for the closeness-reliability tests of Internet2 topology.

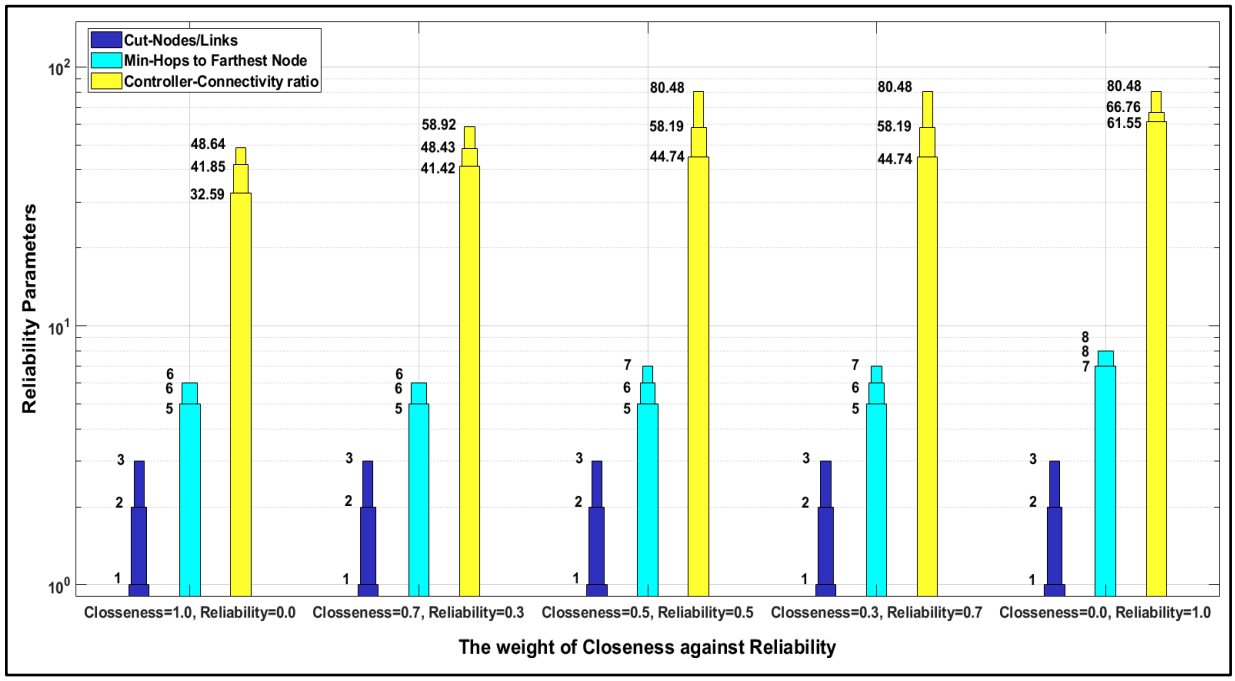


Figure 6.16: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the closeness-reliability tests of Internet2 topology.

1) Discussion of the latency metrics results:

The internet2 network is significantly different from the previous topology (Generated-1). Although it has a very close number of nodes, Internet2 expands over a much wider geographical area than Generated-1. Consequently, all lengths of Internet2 links are much longer than the lengths of Generated-1 links (see Table 6.1). The three-latency metrics (first, second and fourth metrics) in Figure 6.15 above illustrate that the latencies do not have a regular attitude. This is because these latency metrics show high values in the first two tests, then they become lower in the third and the fourth tests. Finally, they go to the largest amount at the last test. They should increase duplex bus ly with decreasing the weight of the closeness. Also, the inter-controller latency behaves abnormally, because it retains a stable value, while it should react in opposition to the previous three metrics. In addition, the average and minimum values of the latency metrics have small variations.

The irregularity in the behaviour of the latency metrics occurs because, the COVN placement algorithm is designed to find the shortest path using the hops count while it discards the length of the link, which shows some vulnerability in the latency results when it is applied on networks like Internet2. ***This result indicates that the third and the fourth tests have the best latency results.***

2) Discussion of the reliability metrics results:

Like Generated-1 network, the reliability metrics of Internet2 network escalates with increasing the reliability weight. Figure 6.16 exposes that the maximum value of the minimum-hops to the farthest node increases in the third test (when the reliability=0.5), while the average and minimum values are stable, which decreases the reliability of the control-path of the farthest switches only. In contrast, it reveals that the maximum, average and minimum values of the controller-connectivity ratio are significantly raised in the third test, which proves the controller reliability to all switches. ***Therefore, it is possible to consider the weight of test three is the optimal weight regarding the reliability of SD-WAN***.

3) Define the optimal weights of closeness and reliability:

As discussed above, the third test with the weights (closeness=0.5, reliability=0.5) produces the optimal latency metrics. Also, it offers the best reliability characteristics for the placing of the physical controllers for Internet2 using the COVN algorithm.

Therefore, the weights of the third test are considered the optimal weights for this network.

C) The weights of India topology: The results of applying the strategy of the second test category are shown in Figures 6.17 and 6.18 below (see the description of the figures in the Section "6.3.2, (A), point 1 and point 2"):

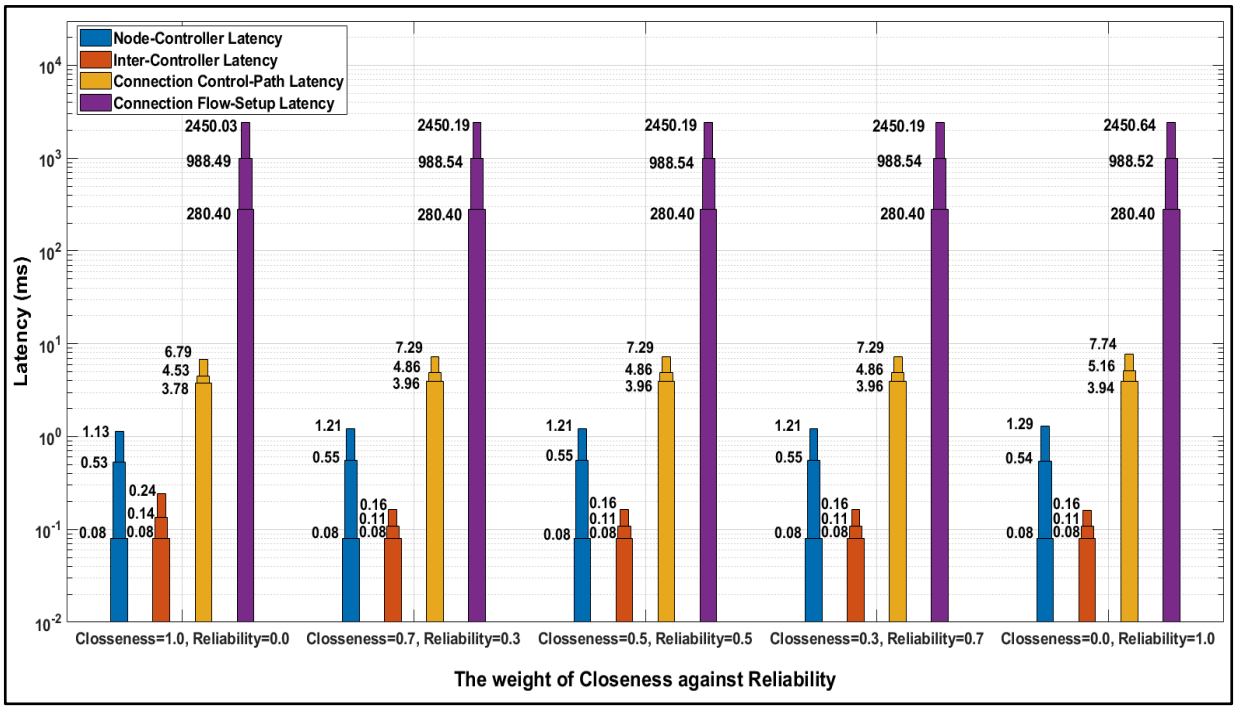


Figure 6.17 : The comparison of the minimum, average and maximum values of the resulted latency metrics for the closeness-reliability tests of India topology.

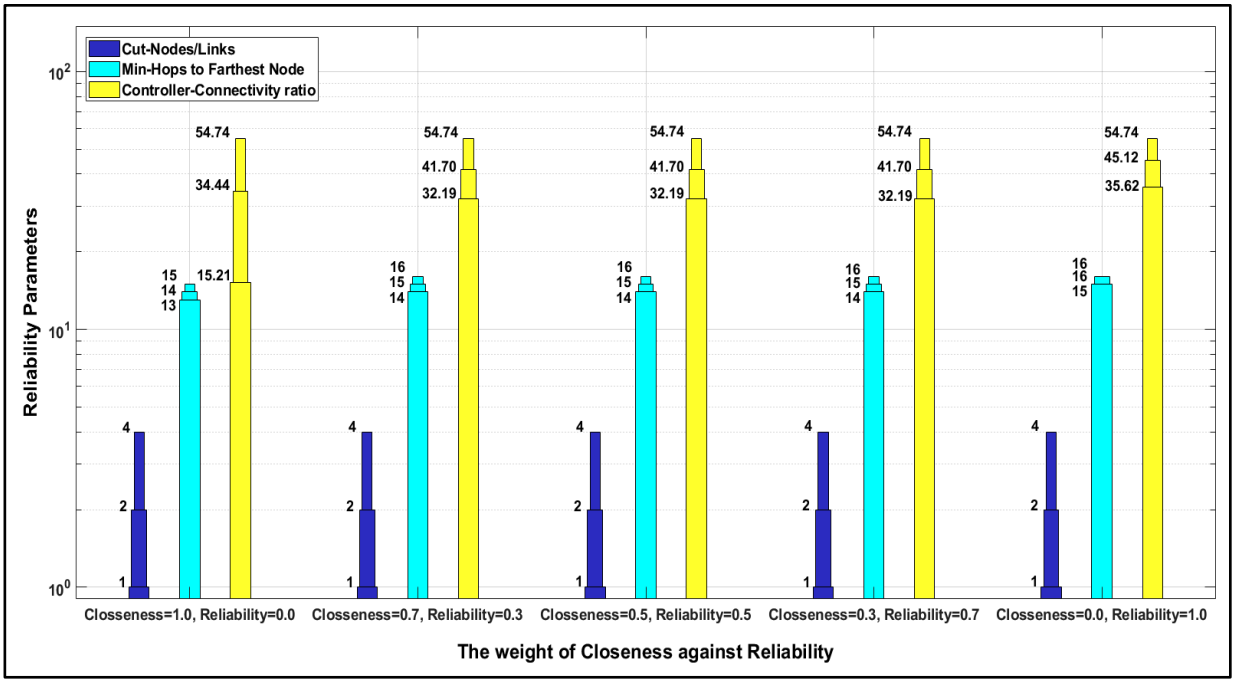


Figure 6.18: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the closeness-reliability tests of India topology.

1) Discussion of the latency metrics results:

India network also extends over the wide geographical area, but it has a large number of nodes (switches). Consequently, the majority of its links length vary between 1-100 Km, while minority links have a longer length (see Table 6.1). The results in Figure 6.17 demonstrate that the latency metrics are raised when increasing the weight of the closeness which is similar to the behaviour of the COVN algorithm when applied on Generated-1topology. The first noticeable increment in these latencies occurred at the second group of bars, after that they show similar values until the last group which again increases these latencies by approximately 40 μs. This result introduces the probability of using the weight of the middle three tests for better latencies or using the weights of the last test if the reliability is improved because the last weights do not demolish the latency metrics.

2) Discussion of the reliability metrics results:

Also, in the India network, the reliability metrics grow with the increase of the reliability weight. Figure 6.18 exposes that the maximum and minimum values, of the minimum-hops to the farthest node increase in the second test (when the reliability=0.3), while the average value is stable, the reliability of control-path only declines at the farthest and nearest switches. In contrast, it illustrates that the average and minimum value of controller-connectivity ratio raises at the last test. This provides a greater improvement for the reliability than optimising the lengths of the maximum and minimum control-path. Therefore, the weights (reliability=1) of the last test is preferred and are considered as the optimal weights for the best reliability.

3) Defining the optimal weights of Closeness and reliability:

From the previous argument, the last test with the weights (closeness=0, reliability=1) produces the acceptable latency metrics. Also, it offers the most reliable characteristics for placing the physical controllers using the COVN algorithm. Therefore, the weights of the last test are considered the optimal weights for India network.

The findings of the second category of tests:

- ***All the previous decisions would not be considered as the optimal decision for all cases and all implementation. The optimal decision would always depend on the preferred parameter for the required placement. While, the decision made in this thesis tries to balance between the latency and the reliability metrics.***

- ***The networks which have the majority of its links length less than 100 Km produces optimal controller placement according to the latency when using the CONV algorithm. However, the resulting placement of applying COVN algorithm on Internt2 leads to acceptable latency metrics. This will be demonstrated more clearly when comparing these results with the result of POCO algorithm in Section 6.5.***
- ***Using the controller placement to improve the network reliability is more effective than using it to improve its latency.***

6.3.3. The Clustering

The third category of tests aims to investigate two objectives, which are: the ability to create balanced clusters; and the effect of increasing the number of clusters on the latency and reliability metrics. The test strategy is performed using the following steps:

- The number of the physical controllers starts with a single controller; the number of controllers is increased by one, each test, until there are six controllers.
- Considering the complete topology as a single virtual cluster, then the number of the clusters is increased by one, until there are six clusters.
- The COVN placement is applied on the network with the optimal weights of closeness and reliability that were identified in previous tests for every topology, as shown in Table 6.4.

Topology	Closeness weight	Reliability weight
Generated-1	0.3	0.7
Internet2	0.5	0.5
India	0.0	1.0

Table 6.4: The optimal weights of closeness and reliability for every topology.

- Performing the clustering according to the node balance and the load balance (requests per second for every switch).
- The maximum, average and minimum values of the latency and reliability metrics are compared for the five tests.
- Investigating if the COVN algorithm can create the balanced clusters.
- Investigating the effect of increasing the number of clusters on latency and reliability metrics.
- Implementing the same steps over the three networks (Generated-1, Internet2 and India), to notice how these weights could differ for every network.

A) The clustering of the Generated-1 topology: The clustering results obtained from applying the COVN algorithm on Generated-1 topology is presented in Table 6.5.

No.	Number of Clusters	Nodes Balance	Load Balance (requests per second)
Test1	1	36	36 25451
Test2	2	18 , 18	18, 18 12857, 12594
Test3	3	12, 12, 12	12, 11 , 13 8206, 8417, 8828
Test4	4	9, 9, 9, 9	10, 9, 9, 8 6645, 6480, 6035, 6291
Test5	5	8,7, 7 ,7 ,7	8, 7, 8, 7, 6 5054, 4729, 5254, 5236, 5178
Test6	6	6, 6, 6, 6, 6, 6	7, 6, 7, 6, 5, 5 4340, 4096, 4540, 4307, 4124, 4044

Table 6.5: The implantation of the COVN placement with a different number of clusters on Generate-1 topology.

Table 6.5 proves the first objective of this test, which is the ability of the COVN placement to produce balanced clusters whether using the number of nodes or the load of the nodes as balancing criteria. Also, it reveals that, when clustering the VN with the intention to balance the load, then the number of the switches differs for every cluster because a load of switches should not be equal in most cases.

The second objective is explored in the comparisons of the latency and reliability metrics of the six tests with a different number of clusters, which are presented in Figures 6.19 and 6.20. The comparison of the latency metrics in Figure 6.19 demonstrates that, the values of the node-controller latency and connection control-path latency decreases with the increasing number of clusters. In contrast, the inter-controller latency raises when increasing the number of clusters. Consequently, the flow-setup latency is affected by both. On the one hand, the connection flow-setup latency is diminished when the effect of inter-controller latency is smaller than the connection control path latency, as notified in the results of tests 1, 2 and 3. On the other hand, the connection flow-setup latency grows when the effect of inter-controller latency becomes larger than connection control path latency, as presented in the results of tests 4, 5 and 6. ***Therefore, the impact of the inter-controller latency becomes greater when the number of controllers is increased***.

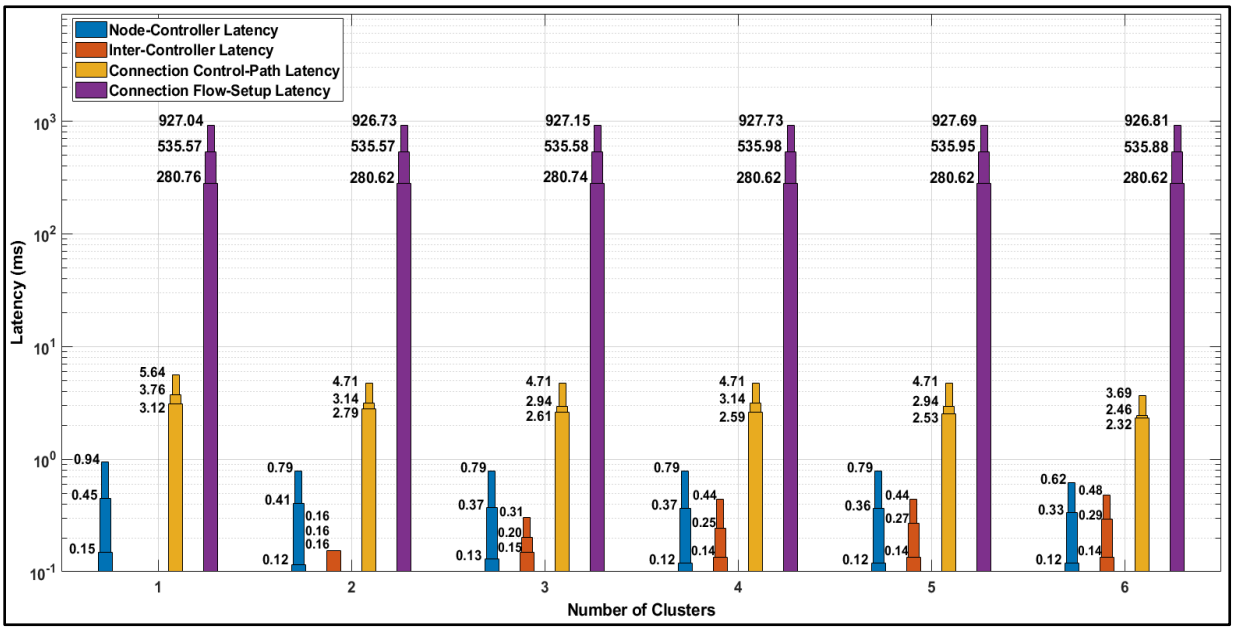


Figure 6.19: The comparison of the minimum, average and maximum values of the resulted latency metrics for the clusters tests of Generated-1 topology.

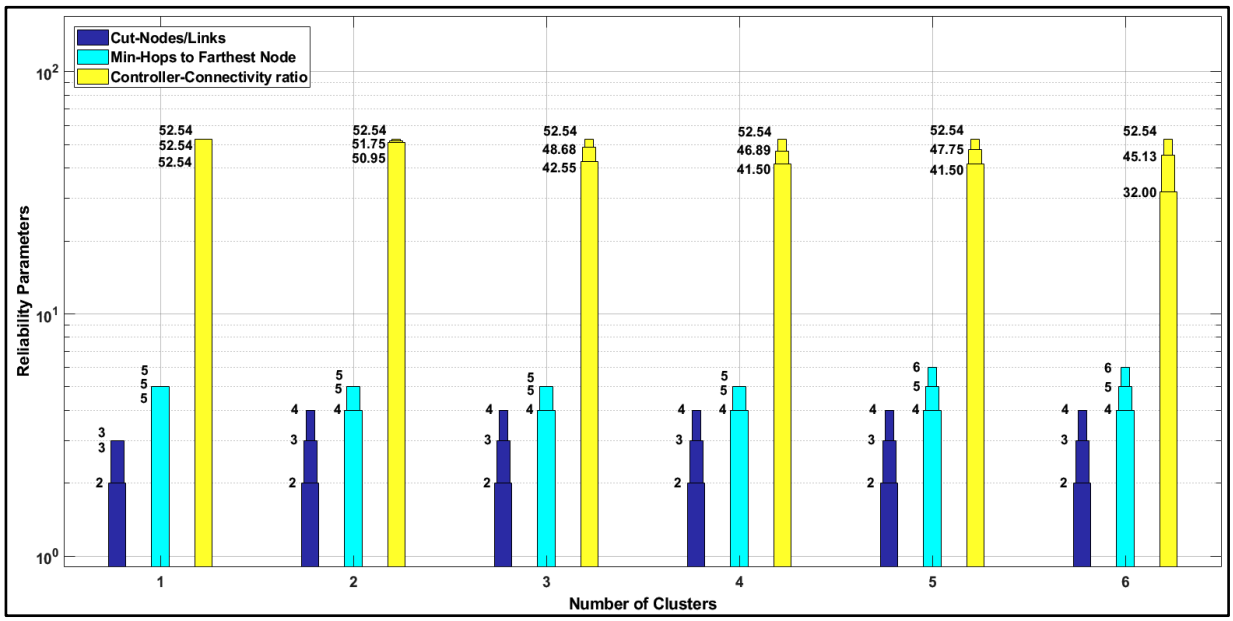


Figure 6.20: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the clusters tests of Generated-1 topology.

The comparison of the reliability metrics in Figure 6.20 illustrates that the maximum value cut-nodes/links metric is increased by one, which optimises the reliability of some switches. Also, ***it shows that the average length of the control-path (average of the minimum-hops to the farthest path) is stable along all tests because all controllers are in a close area***, however, some variance happens when its minimum value increases in the second test and its maximum value rises in the last two tests. ***Finally, the average values of the controller-connectivity ratio is stable with a tiny decrease,*** but its

minimum values go down with increasing the number of the clusters because reliability weight used for this test is 0.7 (highest reliable nodes included) and increasing the controllers leds to additional controller locations being of lower reliability added to the group of the active controller.

B) The clustering of the Internet2 topology: Implementing the COVN algorithm on Internet2 topology with a different number of clusters produces the clustering, which is displayed in Table 6.6.

No.	Number of Clusters	Nodes Balance	Load Balance
Test1	1	34	34 36203
Test2	2	17, 17	1 7, 17 18369, 17834
Test3	3	11, 12, 11	12, 12, 10 12540, 11934, 11729
Test4	4	9, 8, 9, 8	9, 10, 8, 7 9111, 9567, 9323, 8202
Test5	5	7, 7, 6, 7, 7	7, 7, 9, 5, 6, 7513 7046 9607 4954 7083
Test6	6	5, 6, 6, 6, 6, 5	6, 6, 6, 6, 5, 5 6283, 5774, 6020, 6357, 6207, 5562

Table 6.6: The implementation of the COVN placement with a different number of clusters on Internet2 topology.

Like the previous tests the COVN algorithm shows a good balancing ability on both balancing criteria (number or load of nodes), as seen in Table 6.6.

The second objective is investigated through two comparisons which are introduced in Figures 6.21 and 6.22. The latency metrics which are compared in Figure 6.21 show that the node-controller latency and connection control-path latency starts diminishing with the increasing the number of clusters until 4 clusters and goes higher again after that (at 5 and 6 clusters). The behaviour of these two metrics differs from their behaviour on previous topology owing to the effect of the long distances of the link which reduces the decision optimality of COVN placement algorithm. The inter-controller latency grows with increasing the number of a cluster similar to Generated-1 topology metric. Finally, the connection flow-setup latency continues to get a bit larger with the increasing the number of clusters. As a final result, the latency metrics are increased by a small value with the increasing the number of clusters.

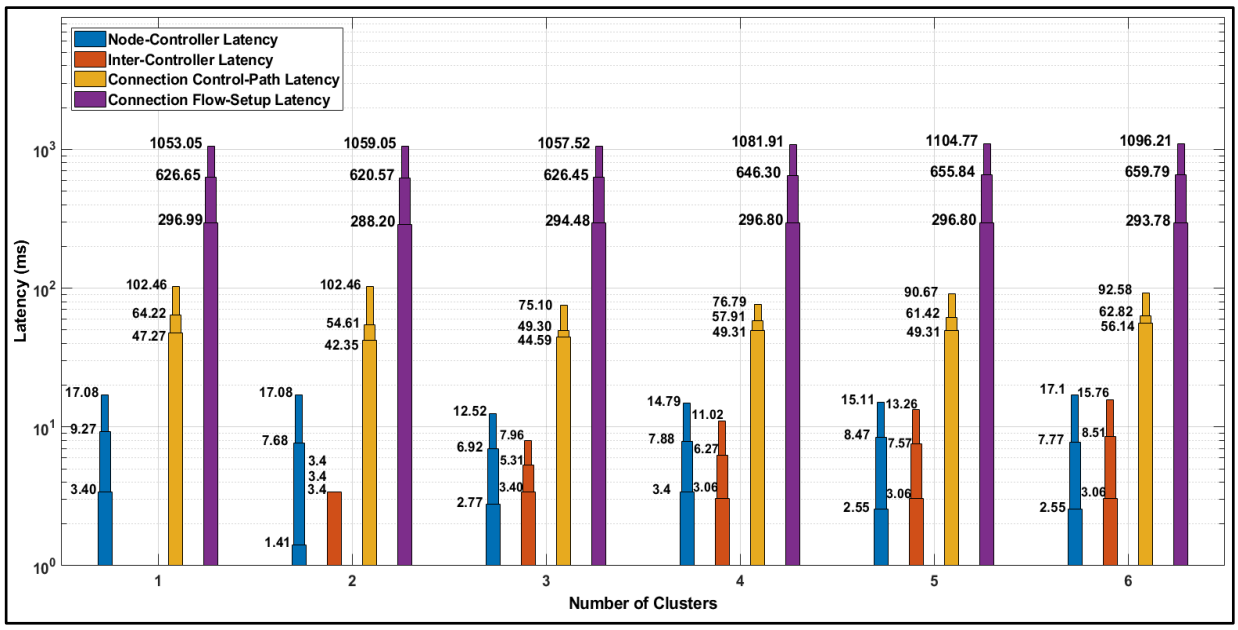


Figure 6.21: The comparison of the minimum, average and maximum values of the resulted latency metrics for the clusters tests of Internet2 topology.

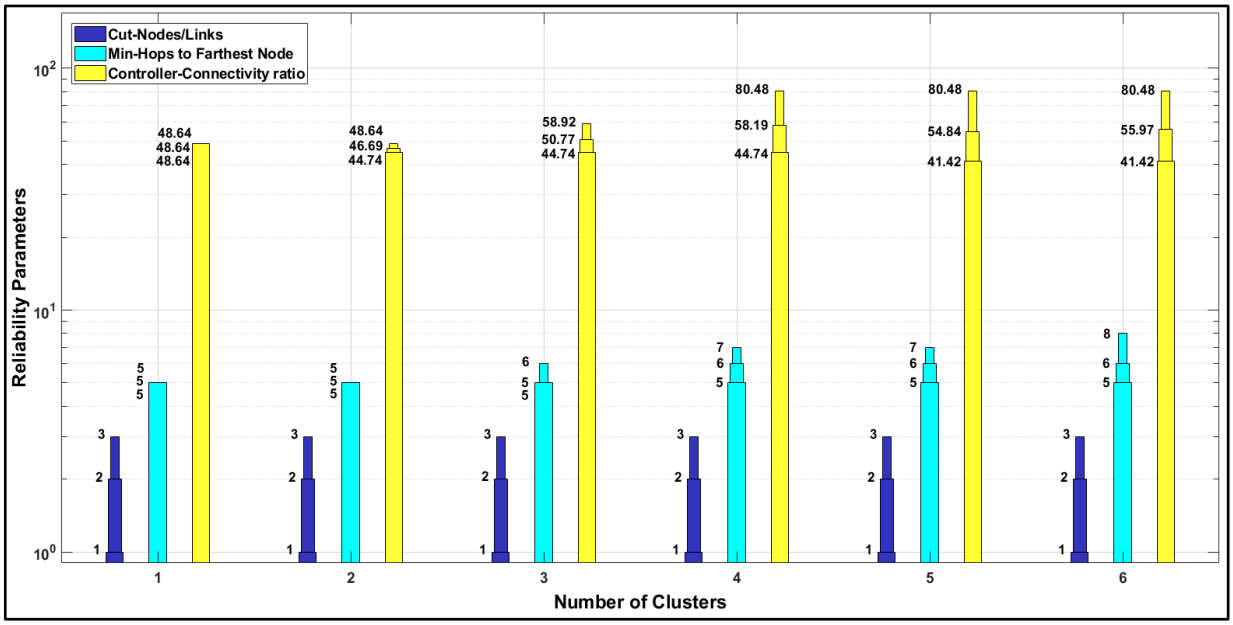


Figure 6.22: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the clusters tests of Internet2 topology.

The reliability metrics of Internet2 are compared in Figure 6.22 and show the very similar behaviour of Generated-1 topology. The length of control-path continues settled only increasing by one in the last three tests. Also, the average value of the controller-connectivity ratio is stable. However, its maximum value expands to 80.48 after the third test. That is because the used reliability weight is 0.5, which means that, the algorithm selected the controllers of medium reliability and when needing to increase the number of controllers the algorithm adds a controller with a higher reliability to the group of

active controllers. ***Therefore, the COVN algorithm provides stable control reliability of SD-WAN with the increment of the number of the clusters.***

C) The clustering of India topology: The third clustering test is implemented on India topology, and the resulted clusters are visible in Table 6.7.

No.	Number of Clusters	Nodes Balance	Load Balance (requests per second)
Test1	1	145	145 16086
Test2	2	72,73	73 , 72 8210, 7876
Test3	3	49,48,48	54, 42, 49 5701, 5279, 5106
Test4	4	37,36,36,36	43, 32, 42, 28 4307, 4062, 4202, 3515
Test5	5	34,32,24,32,23	36, 31, 21, 36, 21 3485, 3968, 2228, 4537, 1868
Test6	6	26, 26, 20, 21, 26, 26	32, 24, 22, 23, 23, 21 3018, 3051, 2343, 2872, 2934, 1868

Table 6.7: The implementation of the COVN placement with a different number of clusters on India topology.

The clustering of the last test shows a similarity to the previous two tests in term of balancing the load of the clusters.

The latency and the reliability metrics of the above six tests are then presented in Figures 6.23 and 6.24 in order to detect the effect of the clustering on these metrics. Figure 6.23 discloses an identical manner to the first test (Generated-1). The node-controller and connection control-path latencies decrease while the inter-controller latency increasers when increasing the number of clusters. Consequently, the flow-setup time stable along the six tests, because the increment in the inter-controller latency is always smaller than the decrease of the connection control path latency in this placement.

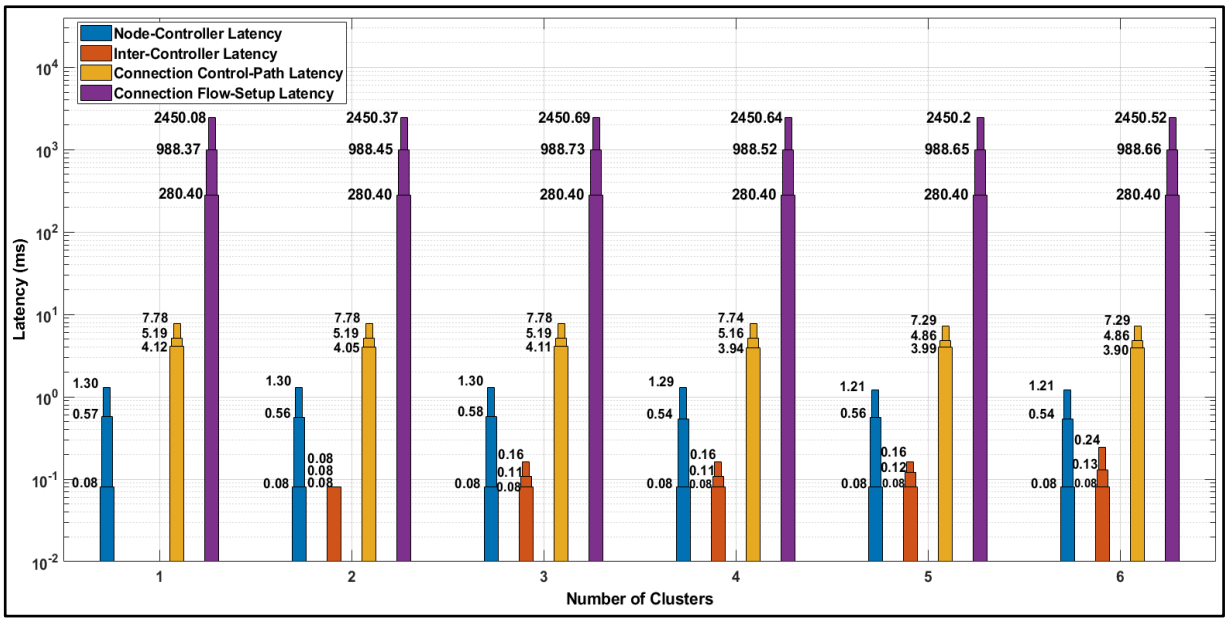


Figure 6.23: The comparison of the minimum, average and maximum values of the resulted latency metrics for the clusters tests of India topology.

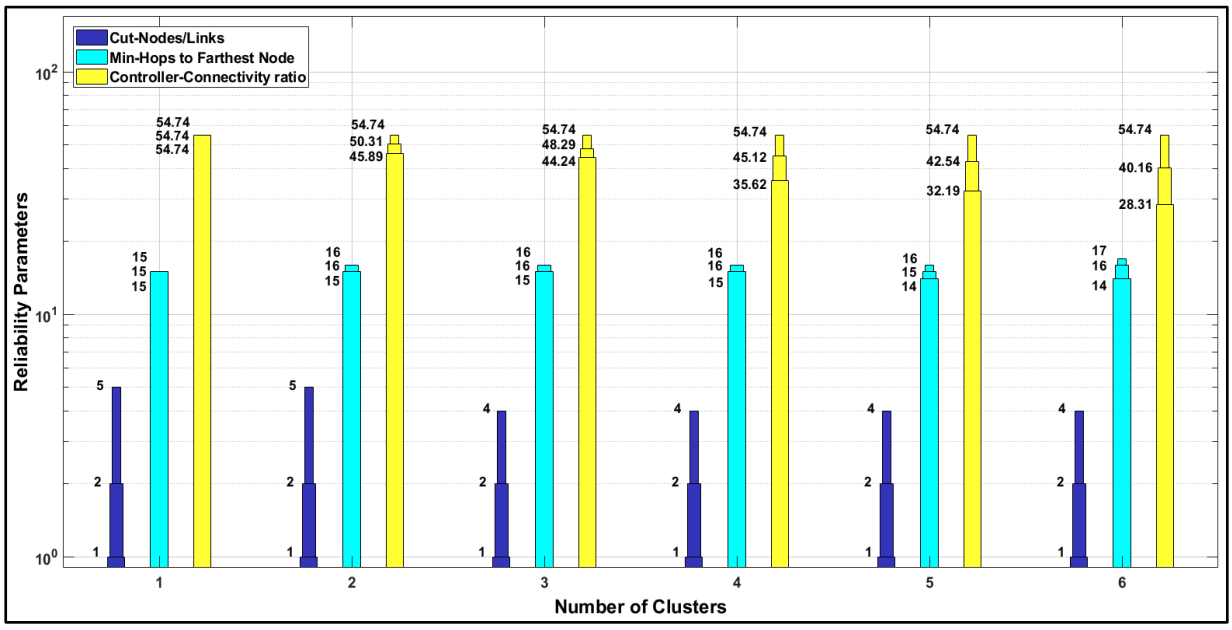


Figure 6.24: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the clusters test of India topology.

Figure 6.24 illustrates that the maximum value of the Cut-nodes/links metric decreases by one after the second test while its average and minimum values are fixed for all tests. Also, it promotes that the average length of the longest control-path (the average value of the minimum-hops to the farthest nodes) keep stable values between 15 and 16. Finally, the maximum value of the controller-connectivity ratio is settled for in all tests while, the average and minimum values of this metric descend with increasing the number of the clusters.

The findings of the third category of tests:

- The COVN placement algorithm efficiently produces balanced clusters.
- The node-controller and connection control-path metrics decrease when the number of clusters is increased.
- The inter-controller metric rises with increasing the number of clusters.
- ***The connection flow-setup metric is affected by the difference between the connection-control path and inter-controller metrics. If the first is larger than the second metric, then the connection flow-setup metric escalates, otherwise the connection flow-setup metric declines. However, the difference between them is minimal because the COVN placement algorithm locates all of the controllers in the close by area.***
- ***The cut-nodes/links and minimum-hops to the farthest node metrics are stable.***
- ***By increasing the number of clusters: The average value of the controller-connectivity ratio is always settled; The maximum value of this ratio increases if the used weight of reliability is less or equal to 0.5 (like Internet2); The minimum value of this metric decreases if the used weight of reliability is larger than 0.5.***

6.3.4. The Controller Failure

The fourth category of tests is performed to prove two objectives. First, the ability of the quick recovery of the controller-failures by the COVN placement algorithm. Second, even though, the VN uses the backup controllers, after using up all of the active controllers, this does not cause significant degradation in the performance of the SD-WAN. This is because, the COVN placement algorithm intends to locate the backup controllers close to the original one, along with considering the reliability of the backup node. The strategy of these tests is as follows:

- The number of the active physical controllers should always be four.
- Considering the complete topology as a single VN, which is partitioned into four clusters.
- The COVN placement is applied on the network with the optimal weights of closeness and reliability identified in the weight tests for every topology, as shown in Table 6.8.

Topology	Closeness weight	Reliability weight
Generated-1	0.3	0.7
Internet2	0.5	0.5
India	0.0	1.0

Table 6.8: The optimal weights of closeness and reliability for every topology.

- The first test starts with the free-failure case, then in every further test, one active controller fails, and it is recovered by a backup controller, until five tests are completed.
- The maximum, average and minimum values of the latency and reliability metrics are compared for the five tests.
- Comparing the locations of the original and backup physical controller.
- Investigating the effect of using the backup controller on latency and reliability metrics.
- Implementing the same steps over three networks (Generated-1, Internet2 and India), to notice how these weights could differ for every network.

A) The controller-failure of Generated-1 topology: Implementing the COVN placement algorithm four times to recover four controller-failures, produce the controller placement which is presented in Figure 6.25.

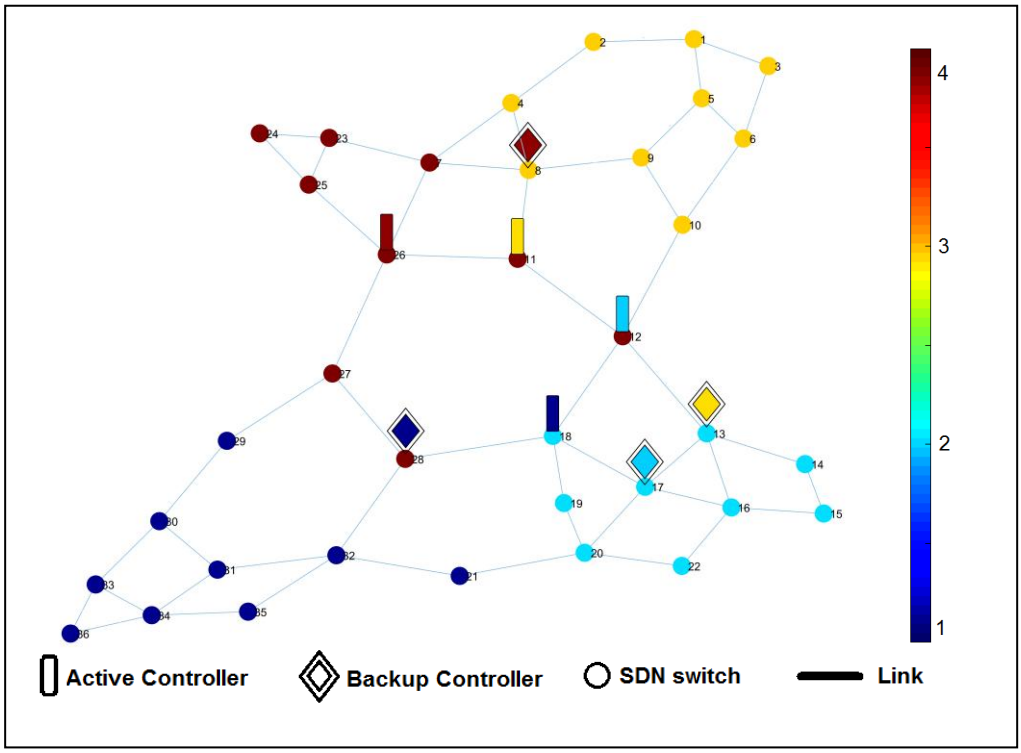


Figure 6.25: The active and backup physical controllers of Generated-1 topology.

The controller placement in Figure 6.25, shows that every active controller (rectangular shape) has been recovered by a backup controller (rhombus shape). The pairs of active-backup controllers could be produced as the displayed pairs, only if the failing and recovering process is implemented in the sequence, which is introduced in Table 6.9, otherwise, different pairs of the active-backup controllers would be created if the failures occur in different sequence.

No.	Failure- Status		
Test1	Failure-Free	Fail-Controllers	X
		Backup-Controllers	X
		Active-Controllers	12, 18, 11, 26
Test2	One Failure	Fail-Controllers	12
		Backup-Controllers	13
		Active -Controllers	18, 11, 26, 13
Test3	Two Failures	Fail-Controllers	12, 18
		Backup-Controllers	13, 28
		Active -Controllers	11, 26, 13, 28
Test4	Three Failures	Fail-Controllers	12, 18, 11
		Backup-Controllers	13, 28, 17
		Active -Controllers	26, 13, 28, 17
Test5	Four Failures	Fail-Controllers	12, 18, 11, 26
		Backup-Controllers	13, 28, 17, 8
		Active -Controllers	13, 28, 17, 8

Table 6.9: The sequence of the controller-failure and recovery process of the five tests on Generate-1 topology.

It is clear that the process of recovering the controller failure does not cost the algorithm additional computational complexity. This is because, the algorithm only assigns the next backup controller to the affected cluster.

The second objective is discovered by comparing the latency and the reliability metrics as offered in Figures 6.26 and 6.27. Figure 6.26 demonstrates that the node-controller latency and connection control-path latency are stable, but the inter-controller latency increases by (0.1 ms) by the last two tests. Consequently, the average value of the connection flow setup latency is increased by (about 1ms) in the last two tests as well, which it is not a significant increment for this latency metric.

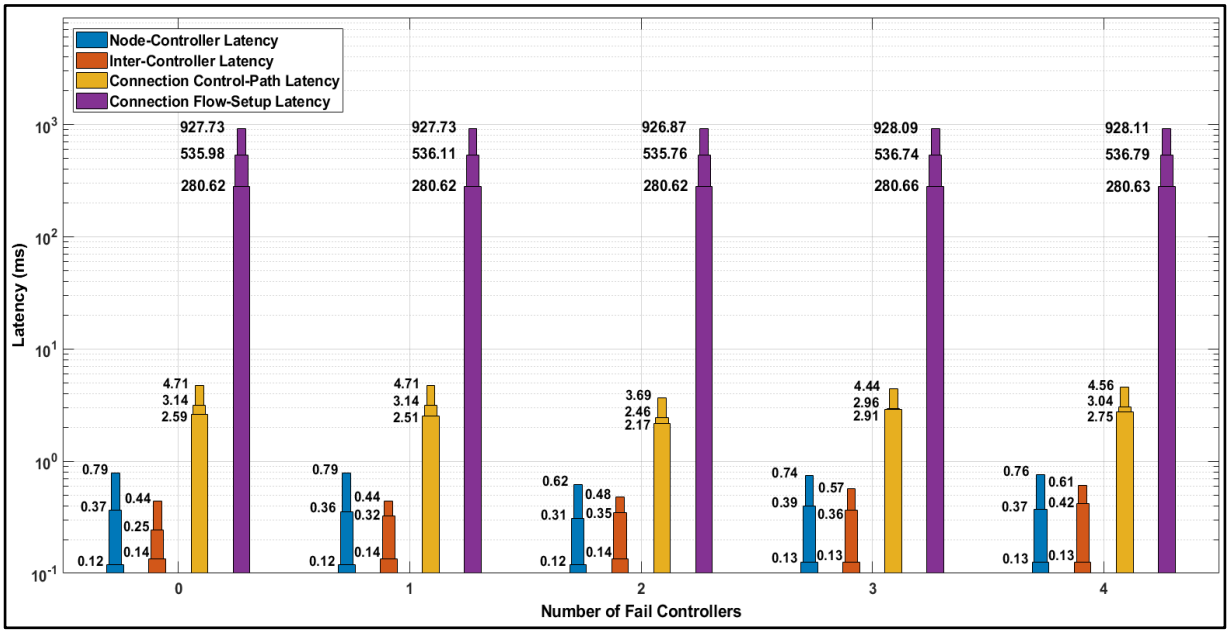


Figure 6.26: The comparison of the minimum, average and maximum values of the resulted latency metrics for the controller-failure tests of Generated-1 topology.

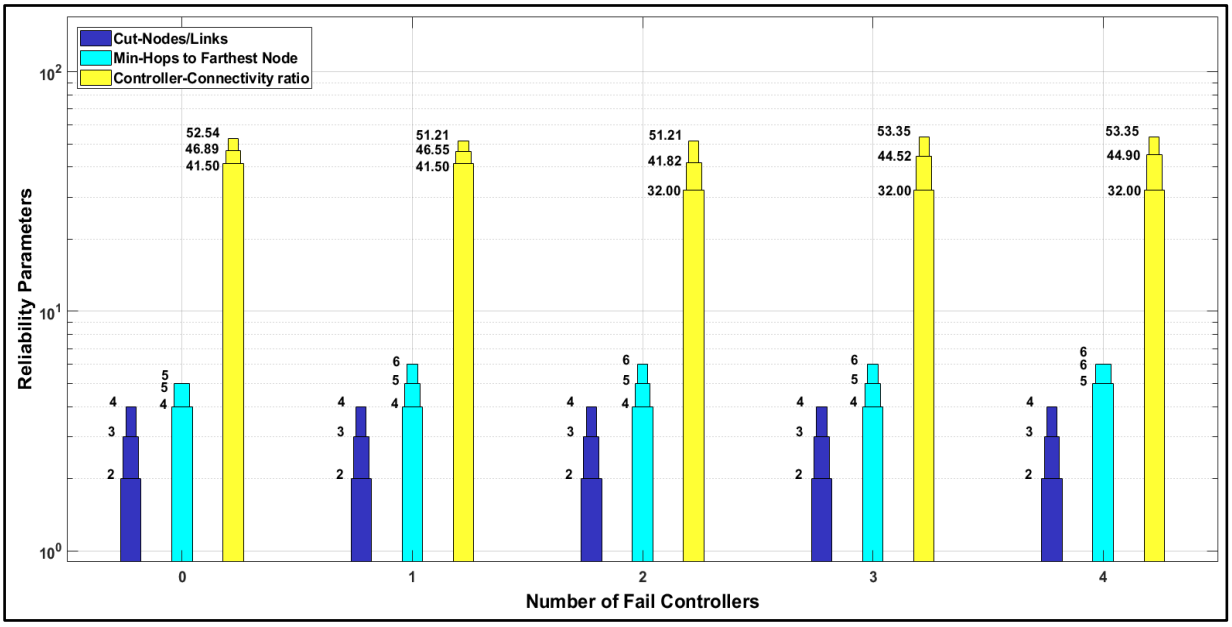


Figure 6.27: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the controller-failure tests Generated-1 topology.

The reliability parameters in Figure 6.27, illustrate a small increment (one hop) in the length of the control-path (min-hops to the farthest node). Also, the maximum and average values of the controller-connectivity ratio stay settled, while its minimum value reduces from 41.50 to 32 in the last three tests.

B) The controller-failure of Internet2 topology: The pairs of active backup controllers that are generated by the controller failure tests, are exhibited in Figure 6.28 below.

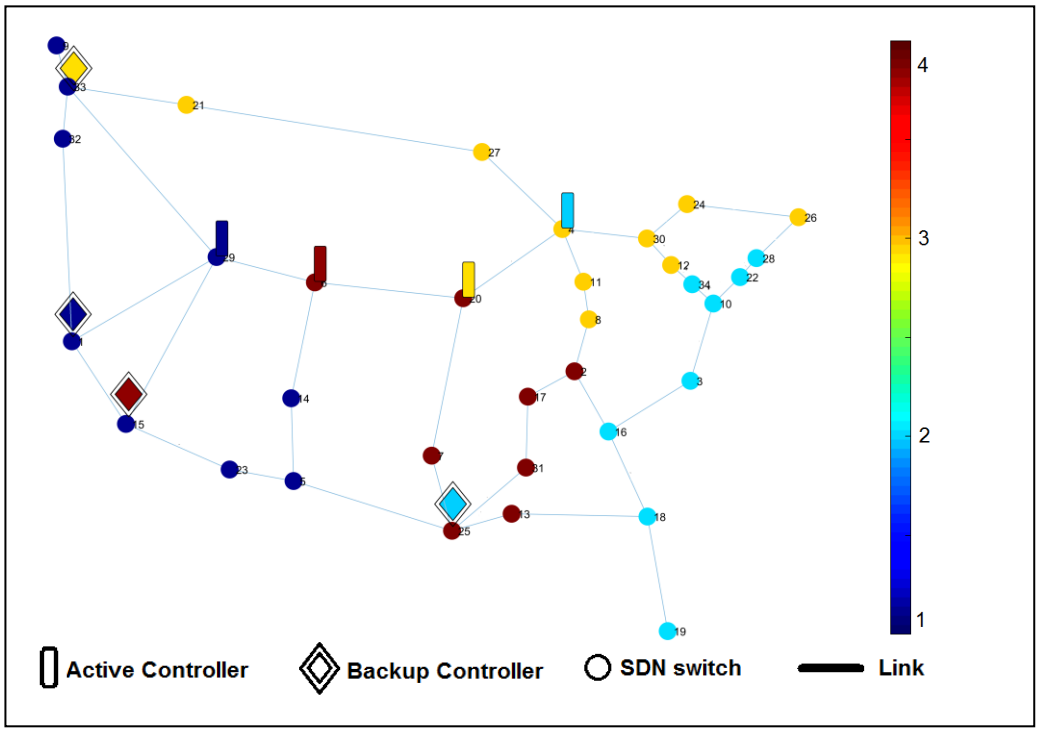


Figure 6.28: The active and backup physical controllers of Intrent2 topology.

The failure tests are performed according to the sequence which is summarised in Table 6.10.

No.	Failure- Status		
Test1	Failure-Free	Fail-Controllers	X
		Backup-Controllers	X
		Active-Controllers	20, 4, 6, 29
Test2	One Failure	Fail-Controllers	20
		Backup-Controllers	25
		Active -Controllers	4,6,29, 25
Test3	Two Failures	Fail-Controllers	20,4
		Backup-Controllers	25, 15
		Active -Controllers	6, 29, 25, 15
Test4	Three Failures	Fail-Controllers	20, 4, 6
		Backup-Controllers	25, 15, 33
		Active -Controllers	29, 25, 15, 33
Test5	Four Failures	Fail-Controllers	20, 4, 6, 29
		Backup-Controllers	25, 15, 33, 1
		Active -Controllers	25, 15, 33, 1

Table 6.10: The sequence of the controller-failure and recovery process of the five tests on Internet2 topology.

Also, the comparisons of the five tests are employed to prove the second objective. Figure 6.29 exposes that all latency metrics increase with increasing the number of fail controllers. The increment this time in the connection flow-setup latency is larger than the increment of the previous tests (Generated-1 topology) due to the longer length of links. It is about 20 ms with every further test of the first four tests, while it is similar for the last two tests.

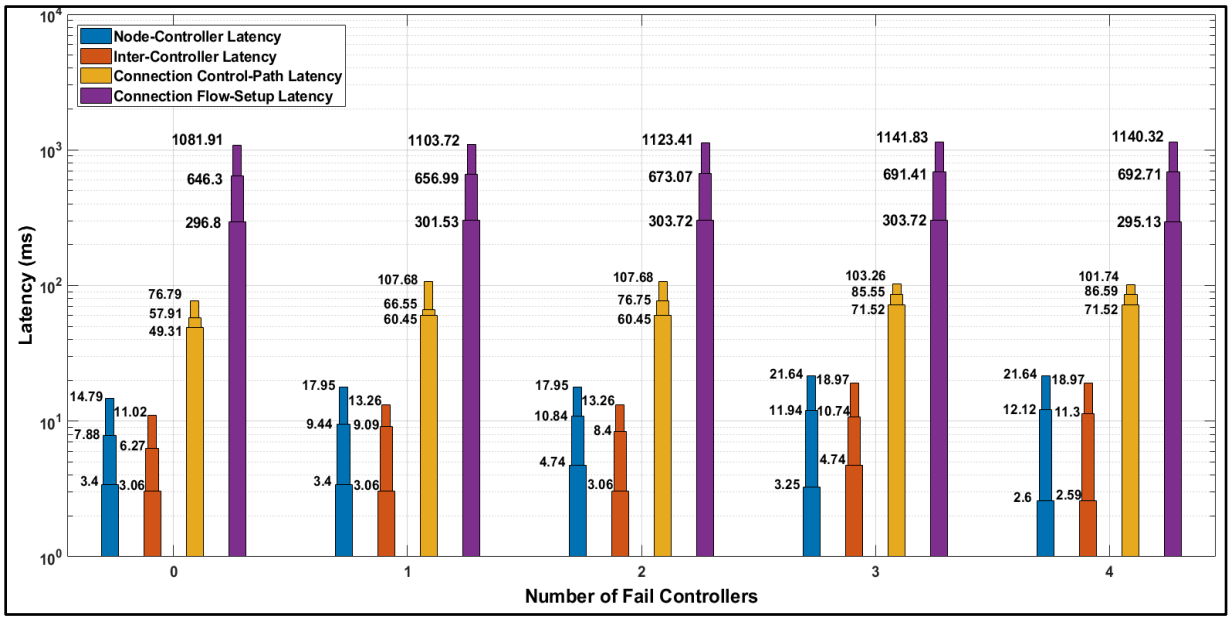


Figure 6.29: The comparison of the minimum, average and maximum values of the resulted latency metrics for the controller-failure tests of Internet2 topology.

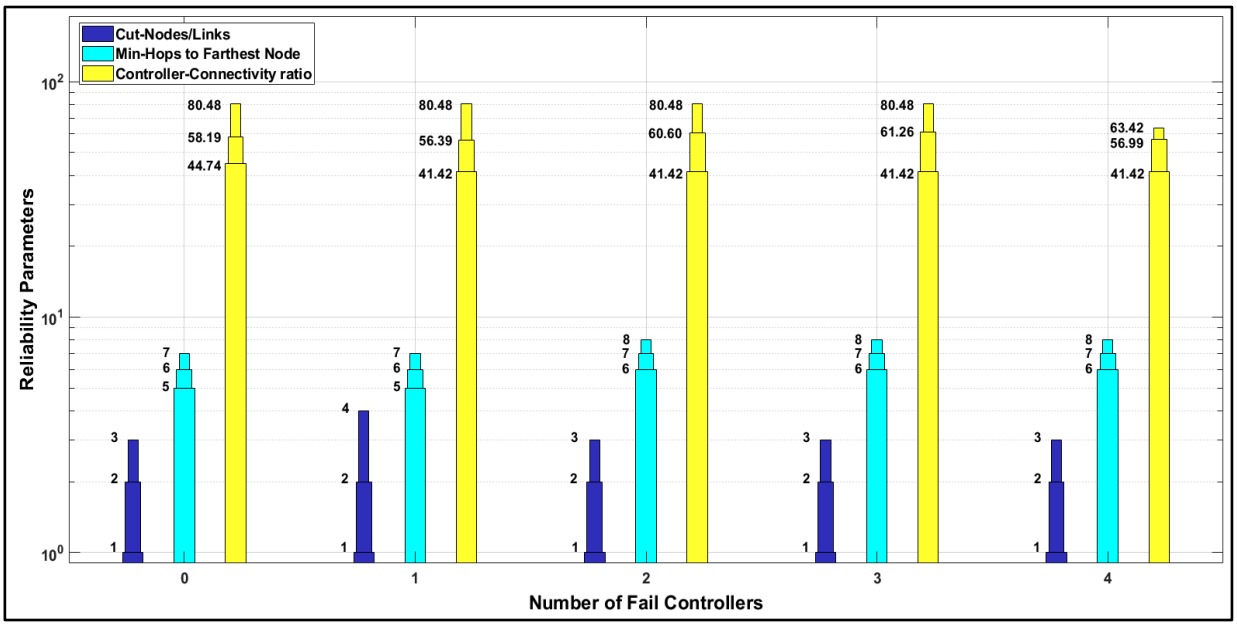


Figure 6.30: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the controller-failure tests of Internet2 topology.

Figure 6.30 expresses that the min-hops to the farthest node mounts by one when three failures happen. Also, it shows that the average and minimum values of the controller connectivity ratio remain the same, while its maximum value falls from 80.48 at fourth test to 63.42 at the fifth test.

C) The controller-failure of India topology: The third controller-failure tests are applied on India network, and the resulted pairs of active-backup controllers are exposed in Figure 6.31.

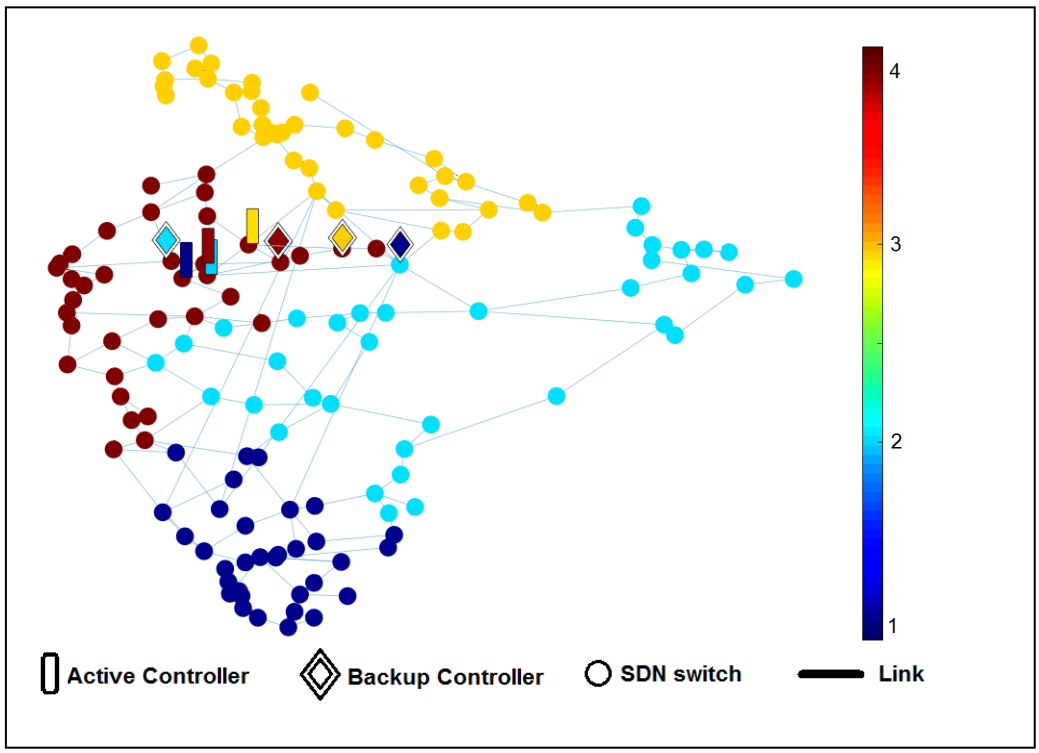


Figure 6.31: The active and backup physical controllers of India topology.

Figure 6.31 simply shows that COVN algorithm spawns the new backup controller (rhombus shape) instead of the active controller (rectangular shape) if the active one is dropped. The failure recovery is archived without re-clustering or recalculating the controller placement in order to obtain a fast recovery.

The sequence of failing the controllers and recovering them using the backup controllers is listed in Table 6.11 below. The table shows that, it starts with failure-free of the controller and ends with falling the four active controllers.

No.	Failure- Status		
Test1	Failure-Free	Fail-Controllers	X
		Backup-Controllers	X
		Active-Controllers	96, 121, 89, 88
Test2	One Failure	Fail-Controllers	96
		Backup-Controllers	72
		Active -Controllers	121, 89, 88, 72
Test3	Two Failures	Fail-Controllers	96,121
		Backup-Controllers	72, 94
		Active -Controllers	89, 88, 72, 94
Test4	Three Failures	Fail-Controllers	96,121,89
		Backup-Controllers	72, 94, 74
		Active -Controllers	88, 72, 94, 74
Test5	Four Failures	Fail-Controllers	96, 121, 89, 88
		Backup-Controllers	72, 94, 74, 95
		Active -Controllers	72, 94, 74, 95

Table 6.11: The sequence of the controller-failure and recovery process of the five tests on India topology.

Next, the comparisons of latency and reliability are discussed to track the changes in these metrics within the controller-failure and to prove the second objective. Figure 6.32 reveals that the latencies heighten in very small values along all tests. This is because the used weight of reliability to apply the COVN placement was the highest value (equals one) for this placement, which orders the controller nodes according to their reliability. Therefore, the next controller node (backup controller) could jump to a remote location from the original one.

The results of the reliability metrics reveal that using the highest weight of the reliability causes a degradation in the controller-connectivity ratio of the backup controllers. Figure 6.33 displays that the controller-connectivity ratio decreases with using more backup controllers. However, the decrement in the average value of this metric is not big; it is about five unit at each test. Also, Figure 6.33 shows that the maximum and average value of the length of the control-path increases by one in the last three tests.

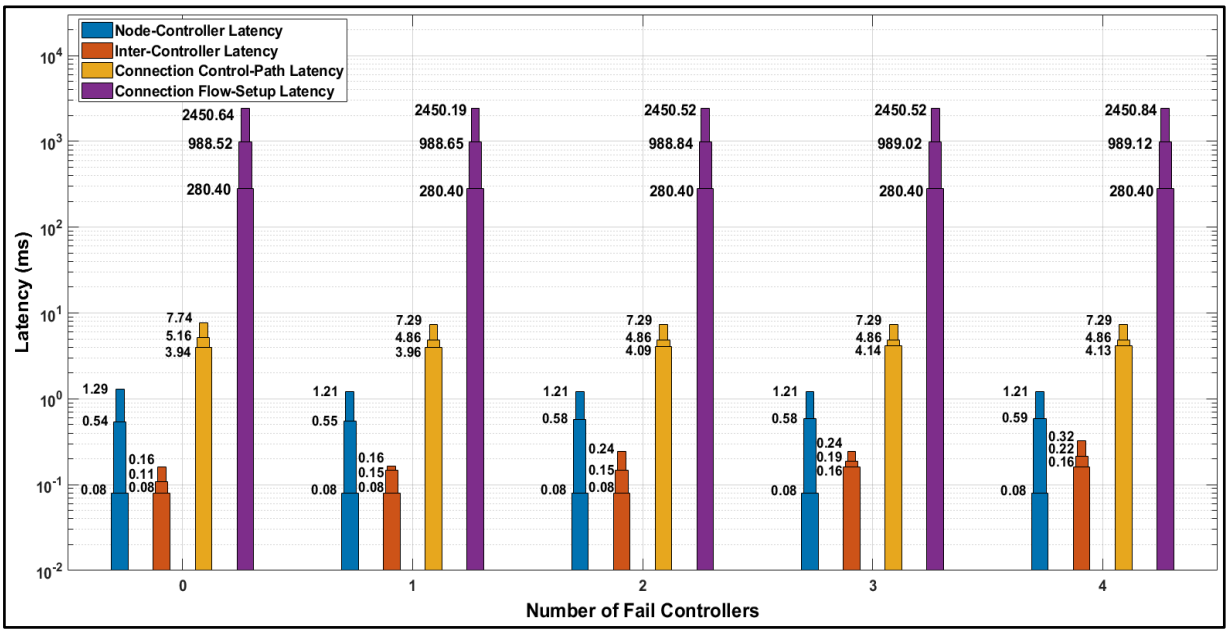


Figure 6.32: The comparison of the minimum, average and maximum values of the resulted latency metrics for the controller-failure tests of India topology.

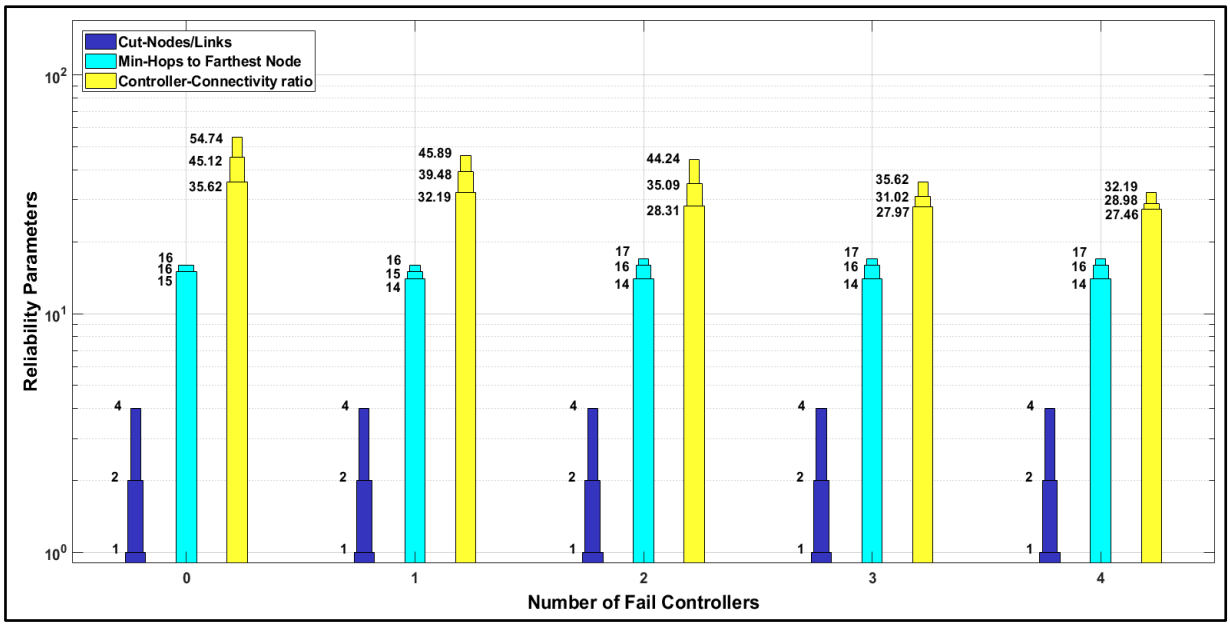


Figure 6.33: The comparison of the minimum, average and maximum values of the resulted reliability metrics for the controller-failure tests of India topology.

The findings of the fourth category of tests (controller-failure tests):

- The COVN algorithm can perform fast recovery for controller failure.
- The latency metrics receive a tiny increment in a microsecond, consequently connection flow-setup latency only increases by milliseconds.
- The reliability metrics are degraded by an acceptable value, which does not demolish the SD-WAN reliability.

Finally, the reasons for not researching (recalculating) the nearest backup controller to the cluster which loses its active controller, are: (1) The failure recovery should be performed as quickly as possible (*it is not about the computation time, but the next back up controller should be configured as a backup controller for all clusters. Subsequently, if the first backup controller is used, then the second backup controller should be configured as the next backup controller for all clusters*); (2) It is preferred that the backup controller is placed in a reliable location and close to other controllers rather than the cluster's nodes. Virtually always (when reliability weight higher than or equals to 0.5) the furthest controller in the controller's list has lower reliability. Also, it always has a longer path to communicate to all other controllers (higher inter-controller latency).

6.3.5. The Changes of the Virtual Slices (Topology or Load)

The objectives for the fifth category of the tests are: First, showing that the COVN placement algorithm can react to the topology changes or the load changes for every VN. Second, proving the ability of the COVN placement algorithm for providing a good controller placement for all VNs of SD-WAN. These tests are performed in the following procedure:

- The number of the active physical controllers should be, three in high load and two in low load.
- Creating two different VNs for every topology:
  1. Horizontal VN: includes only a group of the physical nodes which extends/distributes horizontally for hosting the virtual switches.
  2. Vertical VN: includes only a group of the physical nodes which extends/distributes vertically for hosting the virtual switches.

- For every VN, implementing the COVN placement algorithm on four various statues, which are:
  1. Wide extension: the VN includes all the nodes of this VN.
  2. Shrink extension: the VN include 65% of its nodes.
  3. Heavy load: all nodes of VN work at the highest expected load (requests per second).
  4. Light load: all the nodes of VN work at 65% of the highest expected load.

- The COVN placement is applied on these VNs with the optimal weights of closeness and reliability that were identified in the weight tests for every topology, as shown in Table 6.12.

Topology	Closeness weight	Reliability weight
Generated-1	0.3	0.7
Internet2	0.5	0.5
India	0.0	1.0

Table 6.12: The optimal weights of closeness and reliability for every topology.

- Presenting and comparing the graphs and the clusters' size or load for eight statuses (2 VNs* 4 statuses).
- The maximum, average and minimum values of the connection control-path latency and controller-connectivity ratio are compared for the eight tests to prove that the COVN placement provides a good controller placement in various statuses.
- Implementing the same steps over three networks (Generated-1, Internet2 and India), to notice how these statuses could differ for every network.

A) The VNs of Generated-1 topology: This section illustrates the implementation of the COVN placement to adapt the control layer in reaction to the changes of the VNs. This is performed by presenting the four different statuses of virtual controller placement for every one of the two created VNs, as shown in Figure 6.34 below.

The first four graphs (A-1 to 4), introduce the different controller placements of the first VN, while the last four graphs (B- 1 to 4) exhibit them for the second VN. Graphs (A-1) and (A-2) show that the number of the active virtual nodes are minified from 24 to 16. Consequently, the number of controllers is reduced to optimise the controller utilisation regarding maintaining the same number of nodes for every virtual controller. Graphs (B-1) and (B-2), present a similar procedure but this time, it is for the vertical VN. Both examples demonstrate that the COVN placement achieves a good load balancing in the number of nodes per controller. After that, graphs (A-3) and (A-4), display that the horizontal VN keeps the same topology, but it only changes the load of the switches. Similarly, this occurs in graphs (B-3) and (B-4) of the vertical VN. The last four graphs illustrate that the nodes' load is fairly distributed among the controllers.

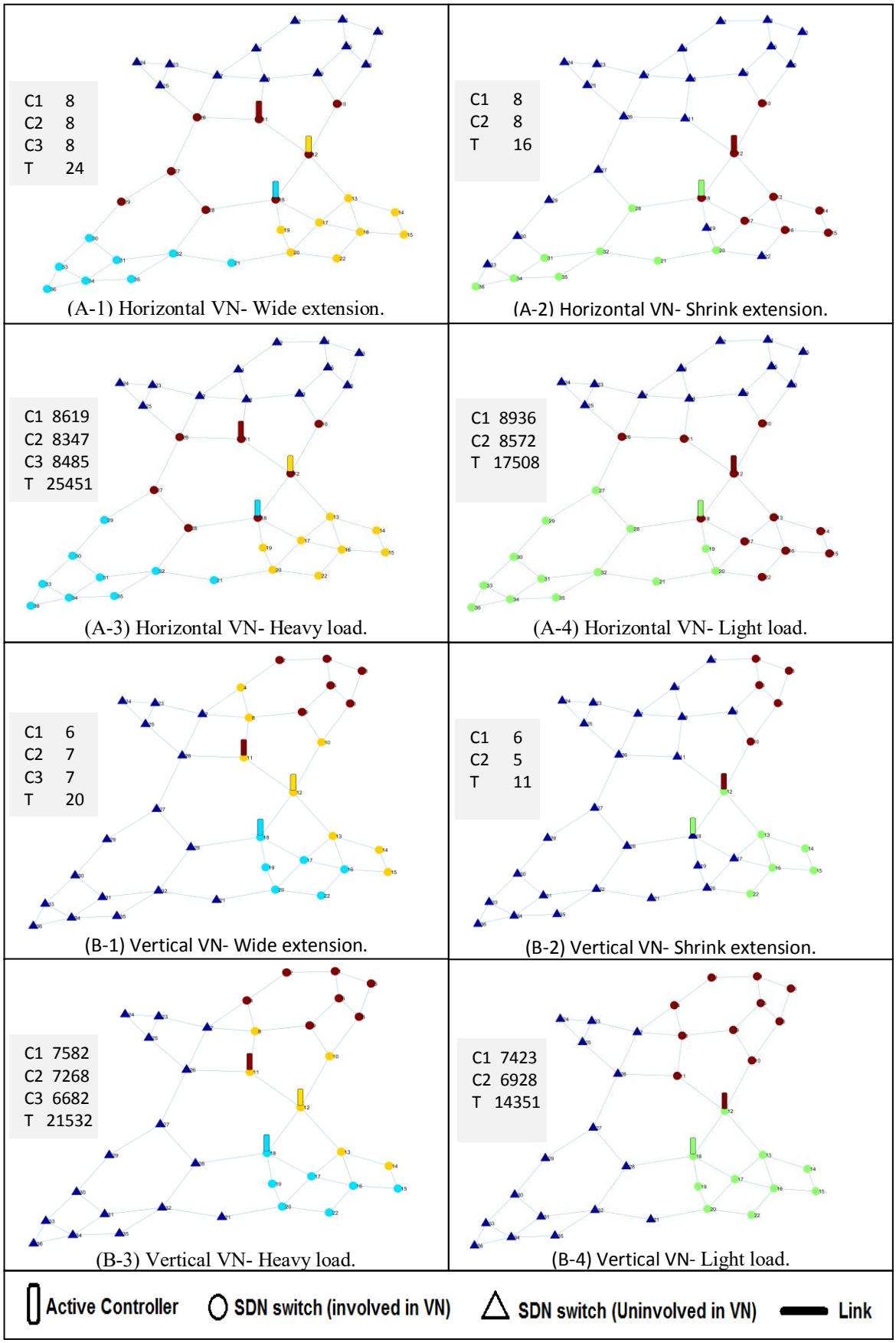


Figure 6.34: The possible changes with the reaction of COVN algorithm for two different VN's of Generated-1 topology.

The next step is comparing some of the resultant metrics of all the controller placements which are presented in Figure 6.34 to investigate the second objective. The connection control-path latency is compared for the eight statuses because it summarises the effect of changing the controller placement, as shown in Figure 6.35 (A). The figure discloses that the average value of the connection control-path latency is stable, while the maximum and minimum values of this latency occupies a small variation (about 1 ms) when changing the controller placement.

Also, Figure 6.35 (B) presents the variation of the controller-connectivity ratio for the eight tests. It shows that this variation is restricted within the acceptable range of connectivity ratio (between 52 and 42).

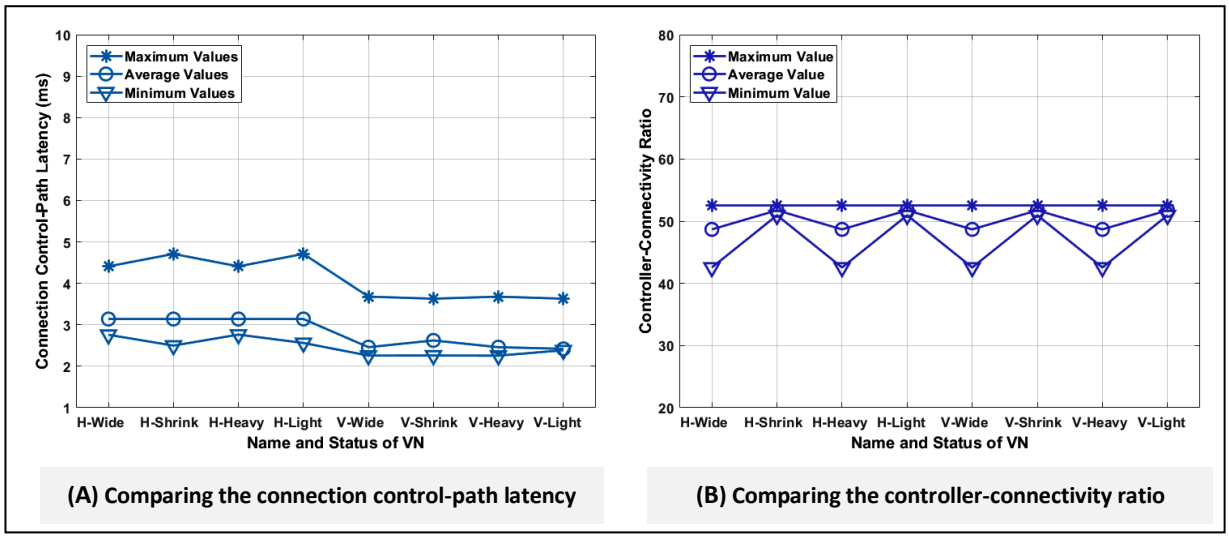


Figure 6.35: The comparisons of some latency and reliability metrics for the eight different statuses of two VNs of Generated-1 topology.

B) The VNs of the of the Internet2 topology: The second implementation of the eight changes of the virtual topology is applied over the Internet2 topology as exposed in Figure 6.36 below.

The eight graphs in Figure 6.36 present the eight cases of the VNs, which are similar to those presented in the previous section. This figure reveals that the COVN placement algorithm acquires a good balance between clusters, whether it is by optimising the number of nodes or the load of the switches.

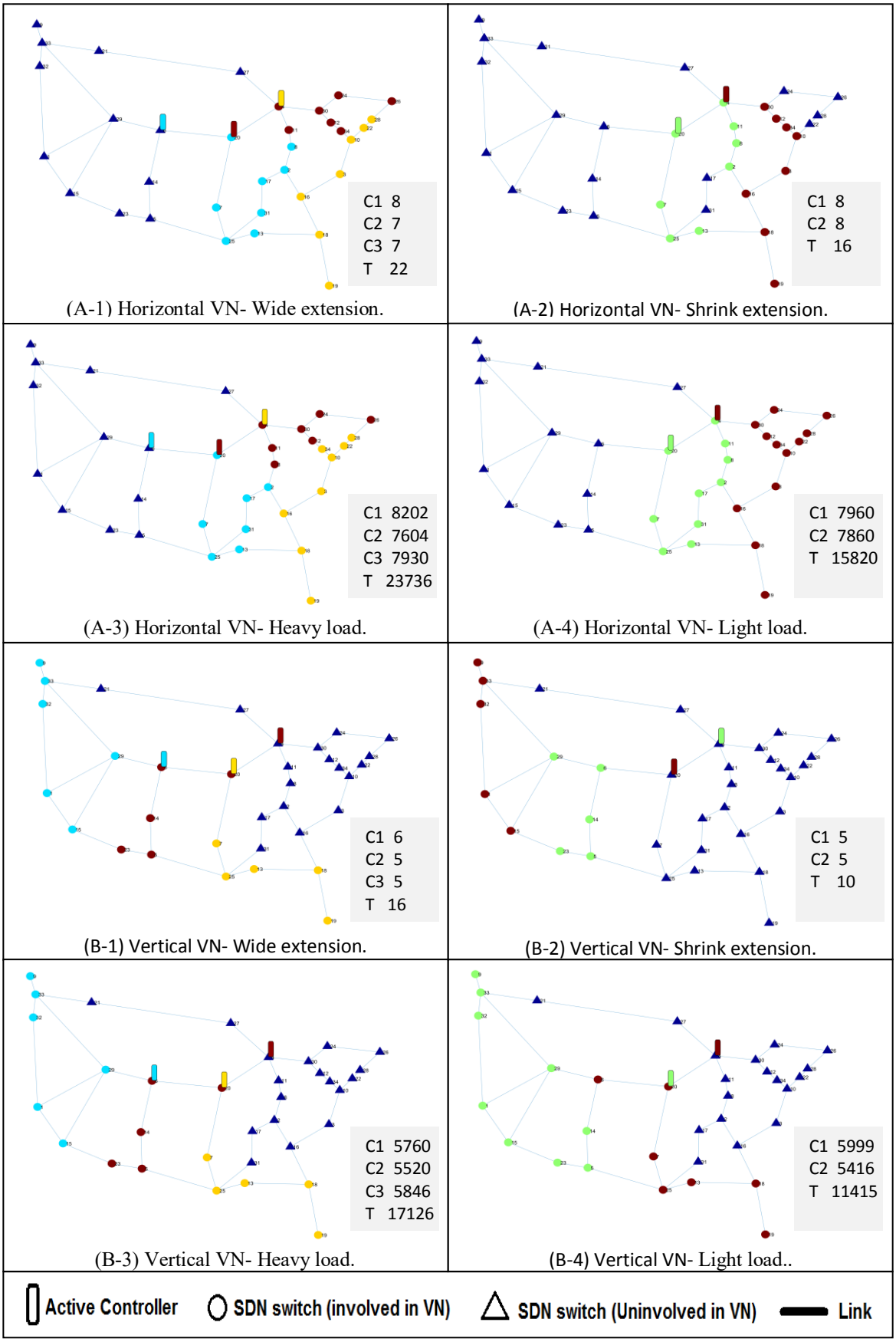


Figure 6.36: The possible changes with the reaction of COVN algorithm for two different VN's of Internet2 topology.

The comparison of the connection control-path latency in Figure 6.37(A), expresses that this latency varies by about 10 to 20 ms up or down when changing the VN status and the placement of the virtual controllers. The amounts of change in latency is not large compared to the long length of links for this topology, which is in thousands of km, whereas the previous topology produces latency changes around 1 ms when the length of links is limited to hundreds of km only. The abnormal behaviour of the average value of this latency at seventh test (V-shrink) resulted from shrinking the nodes of vertical VN towards the network boundaries (right side), which makes them farther from the virtual controllers (see Figure 6.36(B-2) above). However, the increment does not exceed the maximum latency of this topology. For the similar reason, the latency of the first four test (Horizontal VN tests) is lower than the last four tests (vertical VN tests). The nodes of the horizontal VN are closer to the virtual controllers than the nodes of the vertical VN.

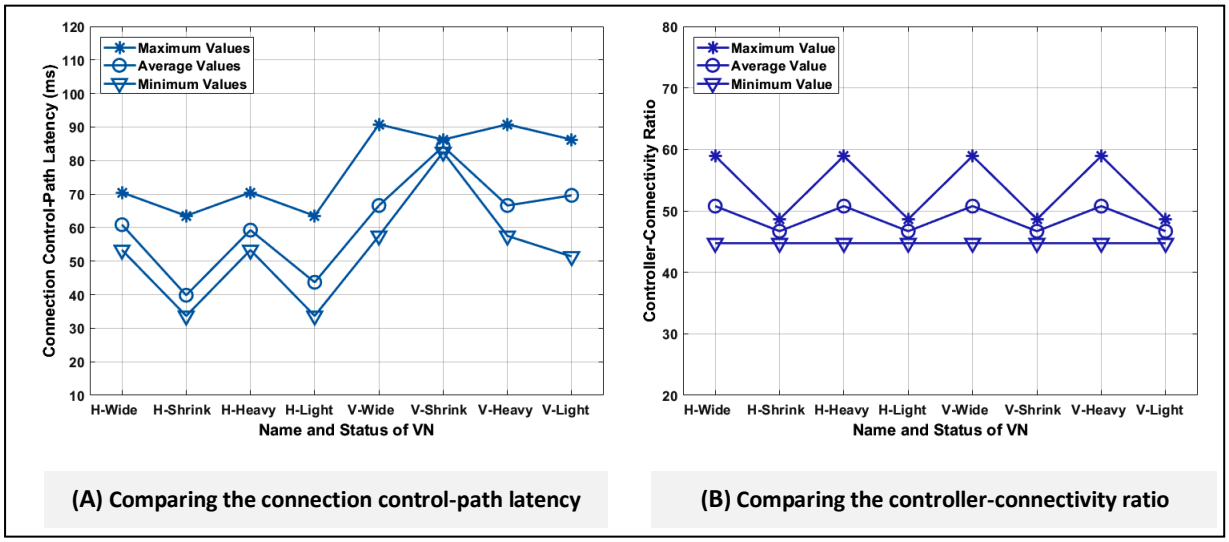


Figure 6.37: The comparisons of some latency and reliability metrics for the eight different statuses of the VNs of Internet2 topology.

Also, Figure 6.37 indicates that the controller connectivity ratio alternates between 45 to 59, which is good connectivity for controller nodes.

C) The VNs of India topology: The third tests of VNs are applied on India network, and the resulted graphs are presented in Figure 6.38 below.

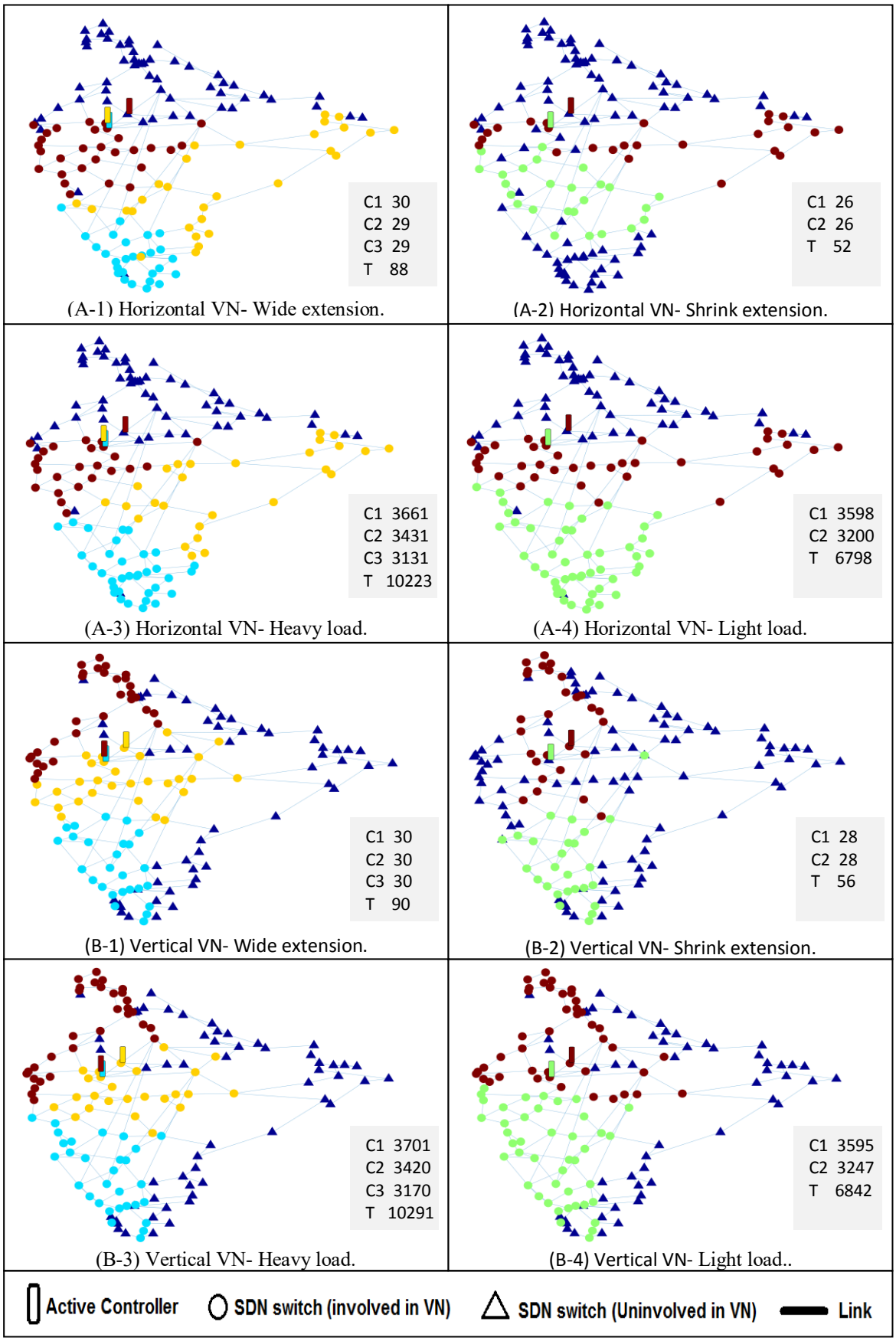


Figure 6.38: The possible changes with the reaction of COVN algorithm for two different VN's of India topology.

Figure 6.38, displays the wide extension, shrink extension, heavy load and light load statuses for the horizontal VN in the first four graphs, then present the same statuses for the vertical VN in the last four graphs. The graphs show that the COVN placement algorithm effectively responds to the changes of the VNs and successfully optimises the controllers' load with the equitable balance of the load. This proves the validity of the first objective of this category of tests.

The second objective is proven based on the comparison below in Figure 6.39. The comparison of the connection control-path latency in Figure 6.39(A), demonstrates that this latency settles with a narrow range of variation (less than 1 ms) during all the occurring changes. However, this latency falls in the second test (H-shrink) because the VN shrinks toward the centre of the network (nearer to the virtual controllers), which reduces the value of this latency. This decrease in the connection control-path latency is beneficial in this case.

Also, the comparisons of the controller-connectivity ratio in Figure 6.39(B), illustrates that their three presented values continue in the same range of connectivity, which regulates the reliability of the network.

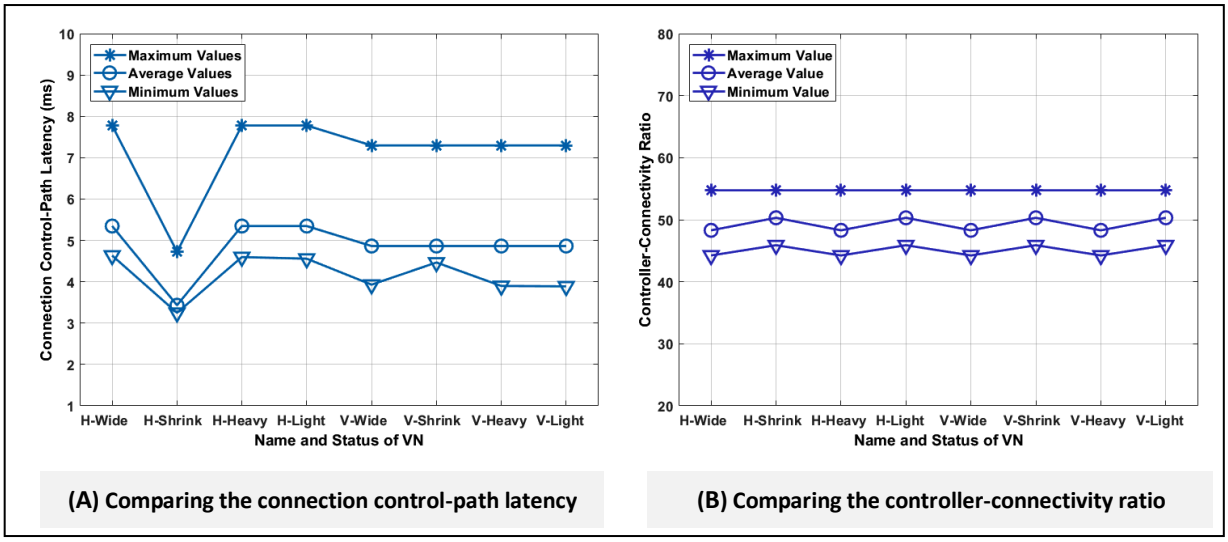


Figure 6.39: The comparisons of some latency and reliability metrics for the eight different statuses of the VNs of India topology.

The findings of the fifth category of tests:

- The COVN placement algorithm can maintain a good balance within a number of nodes or a load of these nodes when re-clustering the VNs.
- The changes of the VN topology produces unavoidable changes in the connection control-path latency and the connection control-path ratio. However, the COVN

placement algorithm is designed to limit these changes to smallest values keeping sufficient performance for all VNs of SD-WAN. This is proved by the results of the three topologies when considering the following issues:

1. The latency variation of Generated-1 and India topologies is about 1 ms, while the variation of the Internet2 topology is about 20 ms due to the long length of these links, which is not related to the performance of the COVN algorithm.
2. The VN gains lower latency when it extends towards the centre of the network, which is not lower than the minimum latency of the complete topology (see India tests).
3. The VN obtains higher latency when it extends towards the boundary of the network, but it does not exceed the maximum latency of the complete network (see the tests of Internet2).

- The controller-connectivity ratio is affected by the number of the used controllers and the number of links of the network, but it is not affected by the location of the VN.

Finally, all the mentioned variations of latency are restricted to its maximum value. The maximum value of latency does not exceed the latency of the connection when its length is equal to half of the network diameter. Also, the controller-connectivity ratio is limited by the minimum value of connectivity for the location of physical controllers, which are in the optimal locations of the network. These restrictions regulate the performance of all VNs at a good level and do not produce unexpected changes that demolish the operation of the VNs.

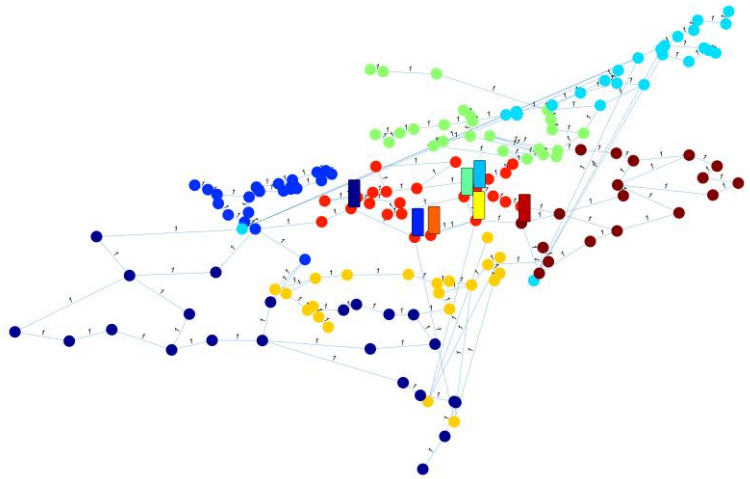
6.3.6. The size of the network

The last category of the simulated tests intends to realize the limitation of the COVN algorithm, through completing three objectives, these are: (1) Confirming the possibility of increasing the number of controllers which are placed by the COVN placement algorithm, if the network size is increased; (2) Exploring the increment of the connection flow-setup latency with growing the size of network; (3) Investigating the increment of the execution time for both parts (part one and part two) of the COVN placement algorithm. The strategy which is used to apply these tests is as follows:

- Choosing a suitable number of controllers for the topology.
- Considering the complete topology as a single VN.
- Applying the COVN placement algorithm with the closeness and reliability weights equal to 0.5 for both.
- Presenting and discussing the resulting controller placement.
- Presenting and discussing the latency and reliability metrics which resulted from applying the controller placements.
- Calculating and viewing the computation time for applying the two parts of the COVN placement on every topology.
- Implementing the previous steps on three topologies which are: Us-Carrier, Generated-2 and Kentucky.

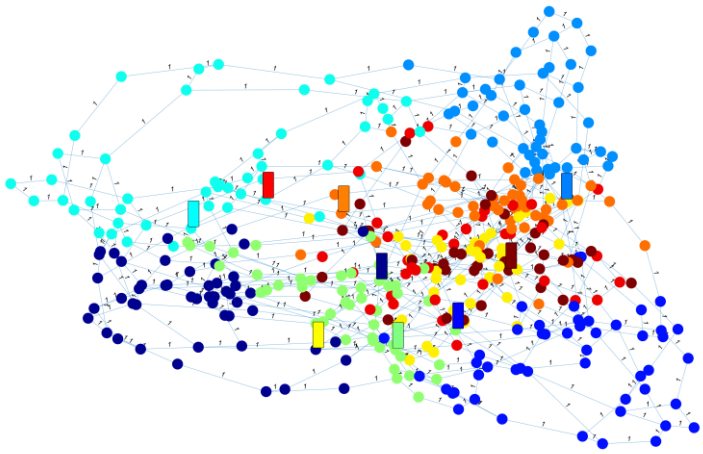
Implementing the COVN placement algorithm on the three topologies produces the controller placements, which are presented in Figure 6.40 below. The graph of US-Carrier network shows that the network is partitioned into seven clusters with a balanced number of nodes (see Figure 6.40(A)). Also, Figure 6.40(B), displays that Generated-2 topology has nine clusters and controllers. Finally, the graph in Figure 6.40(C), presents the controller placement of the Kentucky network. This network is sectioned into eighteen clusters with a balanced number of nodes. However, there are two clusters (C1 and C17) which received only half the number of nodes of other clusters. This is because the nodes of these two clusters are connected linearly and intersected with their neighbour by cut-vertices. The cut-vertex prevents a small number of nodes from moving from the big cluster to the small cluster because disconnecting the cut-vertex from the big cluster creates a separate group of nodes, which generates unbalanced clusters, whether it is added to the original (big cluster) or to the small cluster. This prevents the clustering algorithm from balancing the clusters. Otherwise, the COVN placement algorithm is able to balance all of the clusters.

The number of the controllers is increased from six controllers in the previous three topologies (Generated-1, Internet2 and India), into 7, 9 and 18 controllers in the last three topologies to prove the ability of COVN placement algorithm to increase the number of controllers when the size of the network is increased.



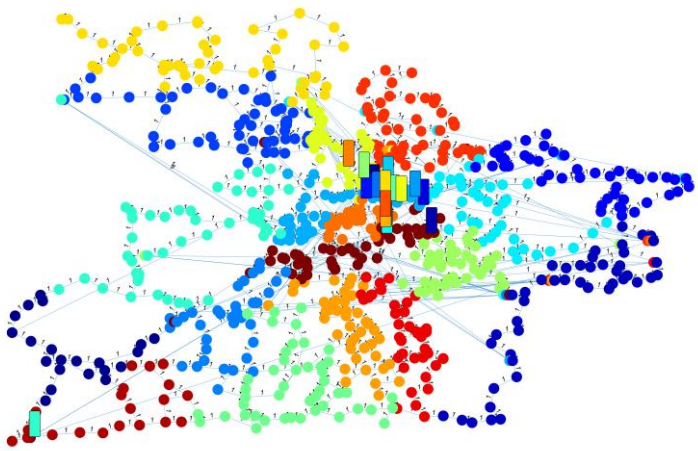
C1	23
C2	22
C3	25
C4	25
C5	21
C6	24
C7	18
Total	158

(**A**) The controller placement of Us-Carrier topology.



C1	49
C2	49
C3	45
C4	48
C5	48
C6	38
C7	48
C8	36
C9	39
Total	400

(**B**) The controller placement of the Generated-2 topology.



C1	21	C10	46
C2	45	C11	43
C3	45	C12	50
C4	50	C13	46
C5	36	C14	31
C6	45	C15	49
C7	44	C16	41
C8	49	C17	21
C9	46	C18	46
Total	754		

(**C**) The controller placement of Kentucky topology.

Figure 6.40: The implementation of the COVN placement algorithm on three different size topologies.

The second objective intends to indicate the increment in the latency and reliability metrics when the network size becomes larger. Figure 6.41 reveals that the increment of the latency metrics is related to two factors: the size of the network and the expansion of the network. However, the expansion of the network has a greater effect in raising the latency metrics. For example, the second test of Generated-2 topology with 400 nodes,

obtains the maximum connection flow-setup latency equal to 1,253.42 ms; while, the first test of Us-carrier with 154 nodes produces 3,014.29 ms for this latency metric due to the larger expansion of the network. Finally, the last test of Kentucky topology, which has the largest network size and expansion demonstrates that the maximum connection flow-setup latency could grow up until 4,780.79 ms.

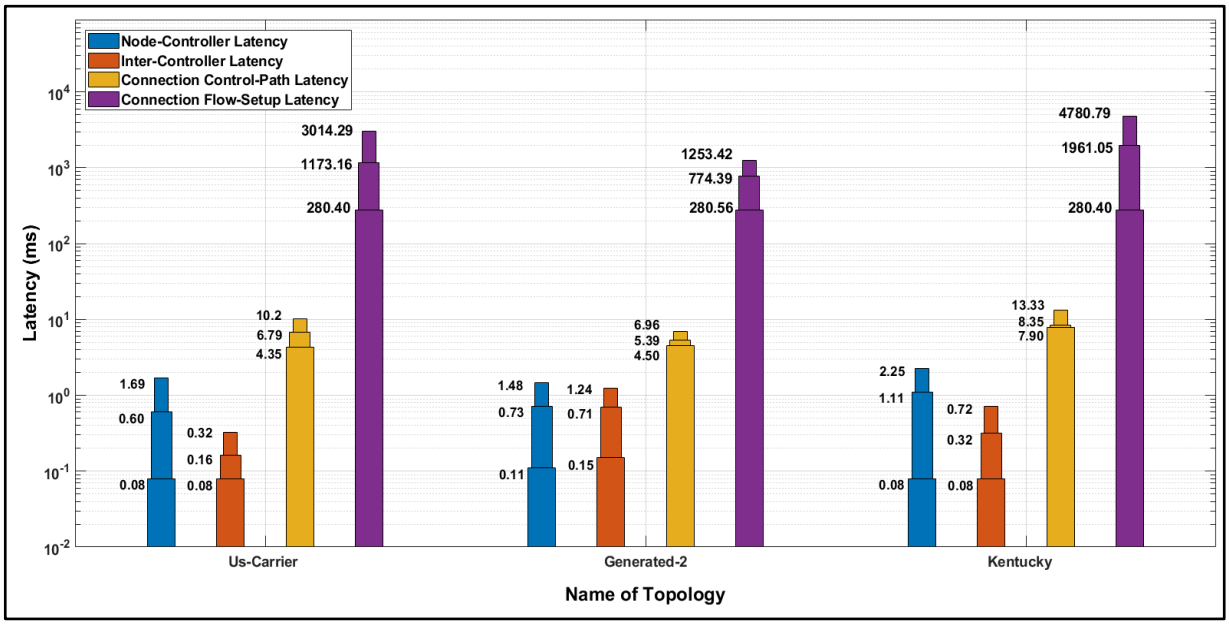


Figure 6.41: The comparison of the minimum, average and maximum values of the resulted latency metrics for three different topologies.

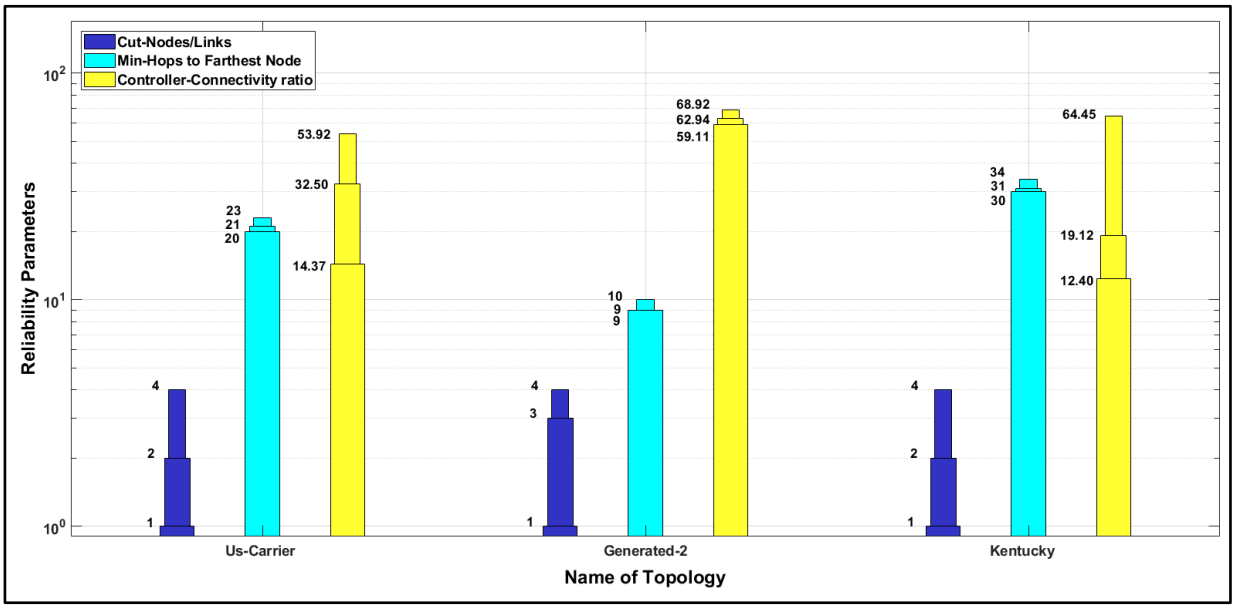


Figure 6.42: The comparison of the minimum, average and maximum values of the resulted reliability metrics for three different topologies.

The reliability metrics, which are presented in Figure 6.42, demonstrate that the length of the control-path (Min-hops to farthest node) is effected by increasing the network size. For instance, the maximum value of this metric equals 23 hops for Us-carrier and

34 hops for Kentucky networks, while it is only ten hops for Generated-2 network. Also, Figure 6.42 proves that controller-connectivity ratio is relevant to the connectivity of the topology. For example, the minimum value of this metric is 14.37 for Us-Carrier topology, while it is 59.11 for Generated-2 topology. However, Generated-2 topology has a larger number of controllers, which reduces the value of the controller-connectivity ratio. The reason for this, is Generated-2 network has more connection between its nodes.

Topology name	Number of controllers	Time of physical controller placement (Second)	Time of virtual controller placement (Second)
Generated-1	6	5.70	4.93
Internet2	6	5.44	4.30
India	6	6.86	5.22
Us-Carrier	7	7.07	6.29
Generated-2	9	20.16	11.25
Kentucky	18	49.53	28.85

Table 6.13: The execution time of the COVN placement algorithm for six different topologies and a different number of controllers.

The execution times of the COVN placement algorithm in Table 6.13 are calculated, to achieve the third objective, which aims to determine the limits of this time. This table discloses that the execution time of the first part (the placement of physical controllers) of the COVN placement algorithm is higher than the second part (the placement of the virtual controllers) of the algorithm. This is corresponding to the design of this algorithm because the second part is applied more frequently than the first part (see Table 4.2 in chapter four). ***Also, the last network (***Kentucky***) in Table 6.13, proves that the COVN placement algorithm has a fast exaction time because it can place eighteen physical controllers on a large network (754 nodes) in only 49.53 seconds and eighteen virtual controllers in 28.85 seconds (see Figure 6.40 (C)).***

The findings of the sixth category of tests:

- The COVN placement algorithm can efficiently increase the number of the placed physical and virtual controllers when the size of the network is increased.
- The latency of VN raises when the size of the network is increased. However, it acquires a higher increment with the network of the wider expansion like Us-carrier and Kentucky.

- The COVN placement algorithm keeps a low execution time even when the size of the network and number of placed controllers are increased.

6.4. The Results of Mininet Emulator

Some tests, which are performed by the COVN simulator, are reapplied on Mininet emulator in order to prove the validity of the simulation results, namely the connection flow-setup latency. These tests are conducted on Mininet to examine this latency between the nearest and farthest pair of the nodes for six different numbers of controllers (from 1 to 6 controllers). The strategy which is used to implement these tests and validate the results of COVN simulator is as follows:

1. Building Generated-1 topology in Mininet and connecting it to the HP VAN SDN controller as shown in Figure 6.43. This is completed by using the programmes which are explained in chapter four (see Section "5.4.4. The Programmes of Implementing the Mininet Developments").

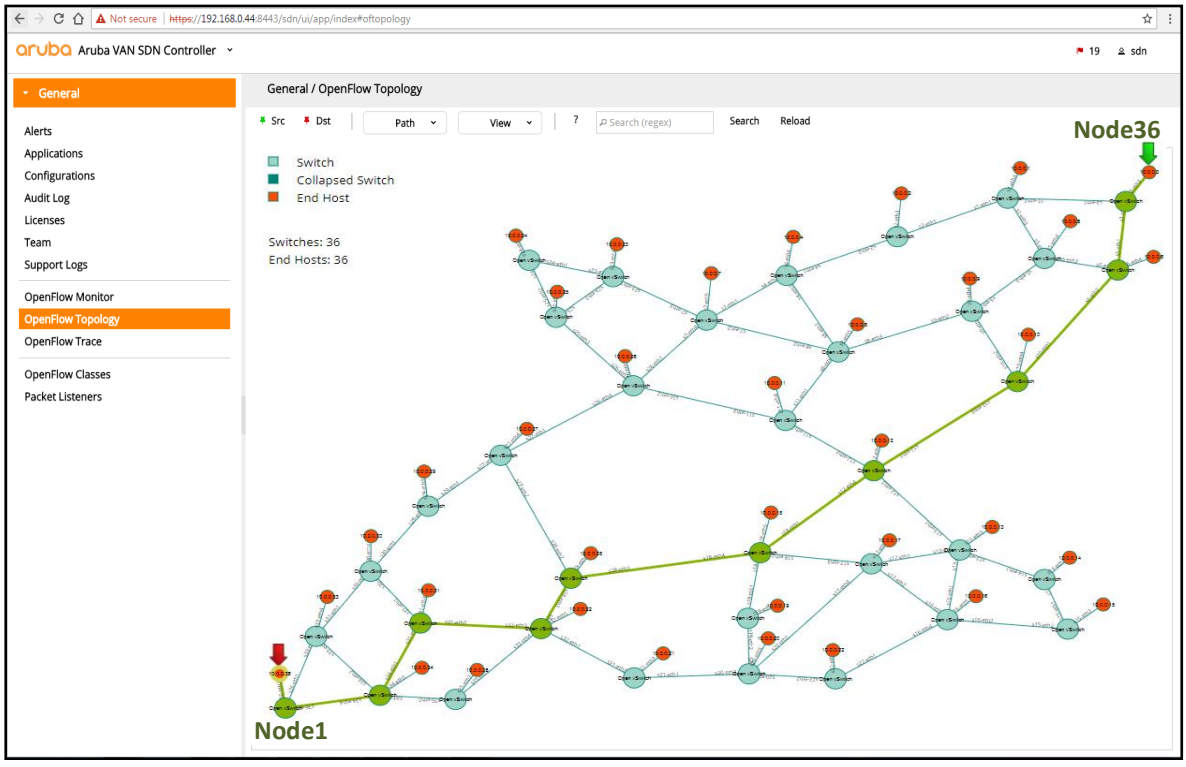


Figure 6.43: The Generated-1 topology in the HP VAN SDN controller interface after executing it on Mininet emulator.

2. Selecting the farthest and nearest two nodes to measure the latency between them. For example, Figure 6.43 presents the path between the farthest two nodes (from node 1 to node 36).

3. Starting the topology with a single controller and single cluster and measuring the connection flow-setup latency of the shortest path between the two farthest and nearest nodes of this network. The latency measurement is performed using the ping tool.

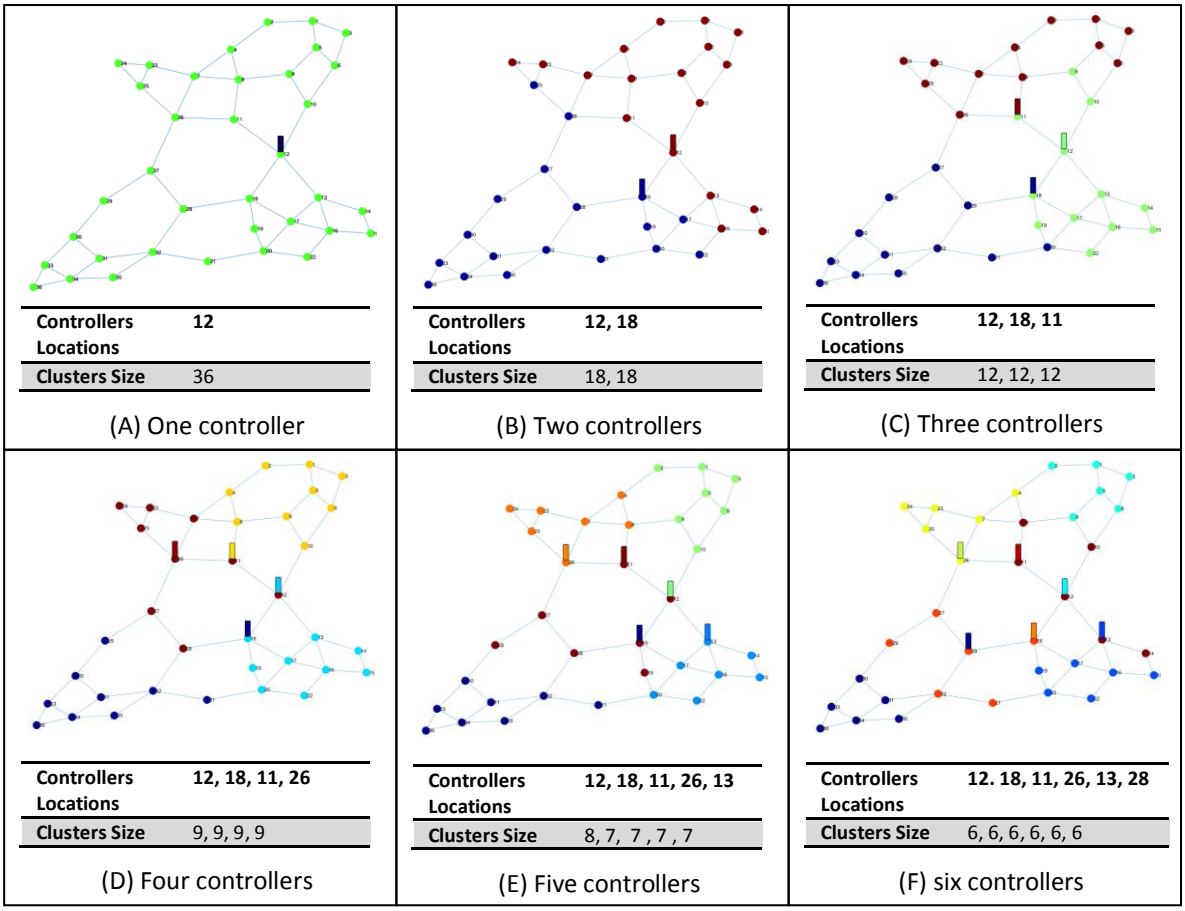


Figure 6.44: The six controller placements which are applied in Mininet tests.

4. Reapplying the previous step on the same network with increasing the number of controllers and their clusters by one controller every test, until six controllers. Figure 6.44 shows the applied controller placements, which are produced by the COVN placement with the weights equal to 0.3 for reliability and 0.7 for closeness.

5. The results of Minimet emulator are compared with the results of the COVN simulator.

Finally, tests are conducted with a different number of controllers to prove the ability of adding the effects of the controllers' location in different six cases by using traffic control (tc) and Network Emulation (netem) tools (see Chapter five Section "5.4.1. The Mininet Limitation and The Developed Solution in Mininet"). The results of the connection flow-

setup latency for Mininet emulator and COVN simulator are introduced in Table 6.14 below.

On the one hand, adding the effect of the control path latency is achieved successfully for all cases according to the results of the last column in Table 6.14 for all cases. On the other hand, the difference between these latencies is not clear in the result because the ping tool can only find the latency in a millisecond, while changing the number and locations of the controllers varies the latency in microseconds. The results in this table represent the average value of 20 tests for every case.

Number of controllers	Src	Dst		COVN Simulator (ms)	Mininet Emulator (ms)
1 Controller					
	1	36	Max. Connection Flow-setup latency	927.035	928
	11	12	Min- Connection Flow-setup latency	280.755	279
2 Controllers					
	1	36	Max. Connection Flow-setup latency	926.725	928
	18	19	Min- Connection Flow-setup latency	280.615	279
3 Controllers					
	1	36	Max. Connection Flow-setup latency	927.145	928
	11	12	Min- Connection Flow-setup latency	280.735	279
4 Controllers					
	1	36	Max. Connection Flow-setup latency	927.73	929
	13	16	Min- Connection Flow-setup latency	280.62	279
5 Controllers					
	1	36	Max. Connection Flow-setup latency	927.685	928
	18	19	Min- Connection Flow-setup latency	280.615	280
6 Controllers					
	1	36	Max. Connection Flow-setup latency	926.805	929
	25	26	Min- Connection Flow-setup latency	280.615	279

Table 6.14: The comparison between the connection flow-setup latency of Mininet and COVN simulator.

Also, the comparison in Table 6.14 demonstrates a matching between the connection flow-setup latency of the COVN simulator and its peer which is measured from Mininet emulator. This proves the validity of the computations and results of COVN simulator.

6.5. The Comparison with results of POCO algorithm

To evaluate the controller placements of the COVN placement algorithm, it is compared to the controller placements of the POCO algorithm. POCO algorithm calculates the optimal placement of controllers by trading-off between six different objectives (Hock *et al.*, 2013). This algorithm and its simulator were built in 2013, and were then developed twice more by (Hock, Hartmann, *et al.*, 2014) and (Lange, Gebert, Zinner, *et al.*, 2015).

The POCO algorithm is chosen for this evaluation because it provides six different controller placements, which are used to optimise the different objectives (see Table 7.15 below). It is the most common algorithm, which is used to evaluate the controller placement in all related work (Lange, Gebert, Spoerhase, *et al.*, 2015) (Naning *et al.*, 2016) (Perrot and Reynaud, 2016) (Liao *et al.*, 2017).

This evaluation is based on comparing three metrics, which are: First, the load balancing; Second, the latency metrics and; Third, the reliability metrics.

The latency metrics of the POCO and COVN placement algorithms are computed for Internet2 topology in two cases, as follow:

A) They are calculated for the ***real*** length of links of the Internet2 topology.

B) They are calculated for the ***minimised*** length of links of Internet2 topology.

Case (A) shows the difference in the performance between the two algorithms on the real topology. While, case (B) is used to prove that the controller placement of the COVN placement algorithm could be more beneficial than the controller placement of the POCO algorithm with small expansion SD-WANs.

The strategy to accomplish these tests is as follows:

- Using four active physical controllers for all tests.
- Considering the complete topology as a single VN, which is partitioned into four clusters for all tests.
- Applying the COVN placement on the network for three statuses:
    1. failure-free;
    2. One-failure;
    3. And three-failures of the controller.

This is to create the statuses of placement which are similar to the placement statuses of POCO algorithm. All of them using weights, which are equal to 0.5 for both closeness and the reliability.

- Applying the POCO algorithm for six different objectives, as shown in Table 6.15.

NO.	POCO Objective	Explanation
1.	Max node to controller (failure-free)	Intends to minimise the latency between the controller and its farthest node, with failure-free of the controller.
2.	Controller imbalance (failure-free)	Intends to balance the number of nodes for all controllers.
3.	Max controller to controller latency (failure-free)	Intends to minimise the maximum inter-controller latency.
4.	Controller-Less nodes (up to two nodes failure)	Intends to optimise the number of paths between the controller and its nodes, up to two paths.
5.	Worst-Latency (failure-Up to 3 controllers)	Intends to minimise the latency between the controller and its farthest node, with considering controller-failure.
6.	Controller imbalance (failure-Up to 3 controllers)	Intends to balance the number of nodes for all controllers, with considering controller-failure.

Table 6.15: The six objectives of the POCO placement algorithm.

- Calculating and comparing the load of the controllers.
- Calculating and comparing the maximum, average and minimum values of the latency and reliability metrics.
- Minimizing Internet2 topology according to Equation 6.1, which makes the maximum length of its links less than 20 km (exactly 19.345 km).

$$Minimized\_length\_internet2 = [\ (1.2 * length\_internet2)\ /100\ ] \qquad (6.1)$$

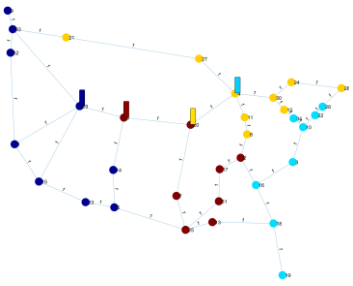
- Implementing the previous steps over the Internet2 network with the minimised length of links.

After the above presentation of the test strategy, the three evaluation comparisons are discussed to determine whether the controller placement of COVN placement algorithm is efficient or not.

First, the comparison analyses the load balancing of the COVN and POCO placements. The three controller placements of the COVN placement algorithm, which are displayed in Figure 6.45 (A, B and C) demonstrate a good load balancing for all controllers. While, the six different controller placements of the POCO algorithm, which are exhibited in Figure 6.45 (from D to I), introduce a poor load balancing, except the graph (E) where the main objective is balancing the load of the controllers. Also, COVN placement algorithm provides the load balancing according to a load of nodes (requests of the flow-

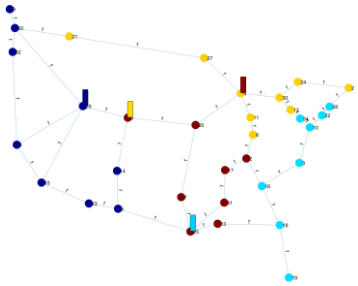
rules per second), while POCO algorithm can only balance the load according to the number of nodes.

The Three COVN placements



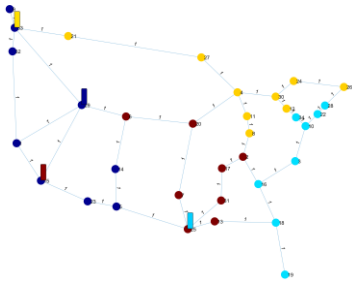
Controllers Locations	20, 4, 6, 29
Clusters Size	9, 8, 9, 8

(A) Failure-free



Controllers Locations	4, 6, 29, 25
Clusters Size	9, 8, 9, 8

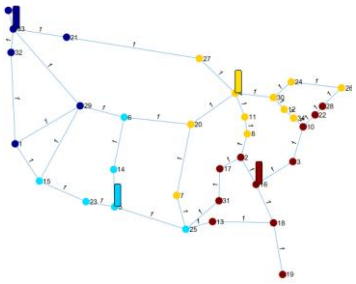
(B) One fail controller



Controllers Locations	29, 25, 15, 33
Clusters Size	9, 8, 9, 8

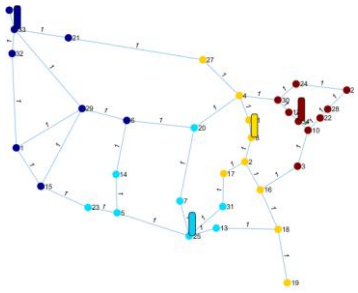
(C) Three fail controllers

The Six POCO placements



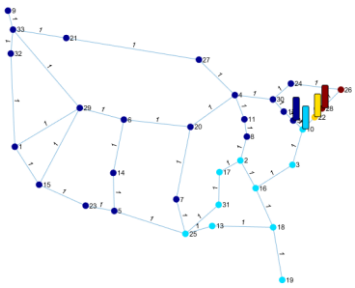
Controllers Locations	33, 5, 4, 16
Clusters Size	6, 6, 11, 11

(D) Max. node to controller



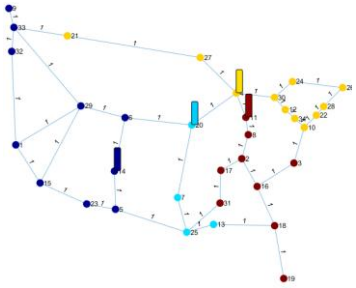
Controllers Locations	33, 25, 8, 34
Clusters Size	8, 8, 9, 9

(E) Controller imbalance



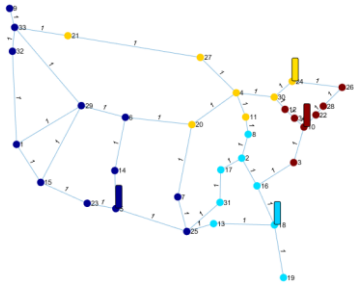
Controllers Locations	34, 10, 22, 28
Clusters Size	21, 10, 1, 2

(F) Max. inter-controller



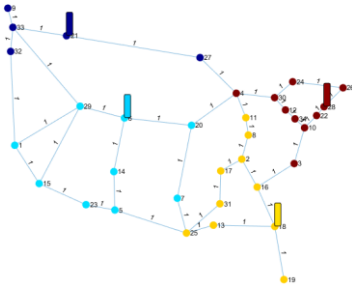
Controllers Locations	14, 20, 4, 11
Clusters Size	10, 4, 11, 9

(G) Worst-latency, three fails



Controllers Locations	5, 18, 24, 10
Clusters Size	12, 8, 7, 7

(H) Controller imbalance, three fails



Controllers Locations	21, 6, 18, 28
Clusters Size	5, 9, 10, 10

(I) Controller-less nodes

Figure 6.45: All controller placements, which are produced by the COVN and POCO placement algorithms to be compared together.

From above the COVN placement has better load balancing capabilities than POCO placement.

The Second comparison matches the latency metrics in two cases, as below:

A) Latency metrics for real length of links of Internet2 topology

In this section, the latency metrics of the two algorithms are compared, when there is no controller failure (see Figure 6.46), and when there is a controller failure (see Figure 6.47).

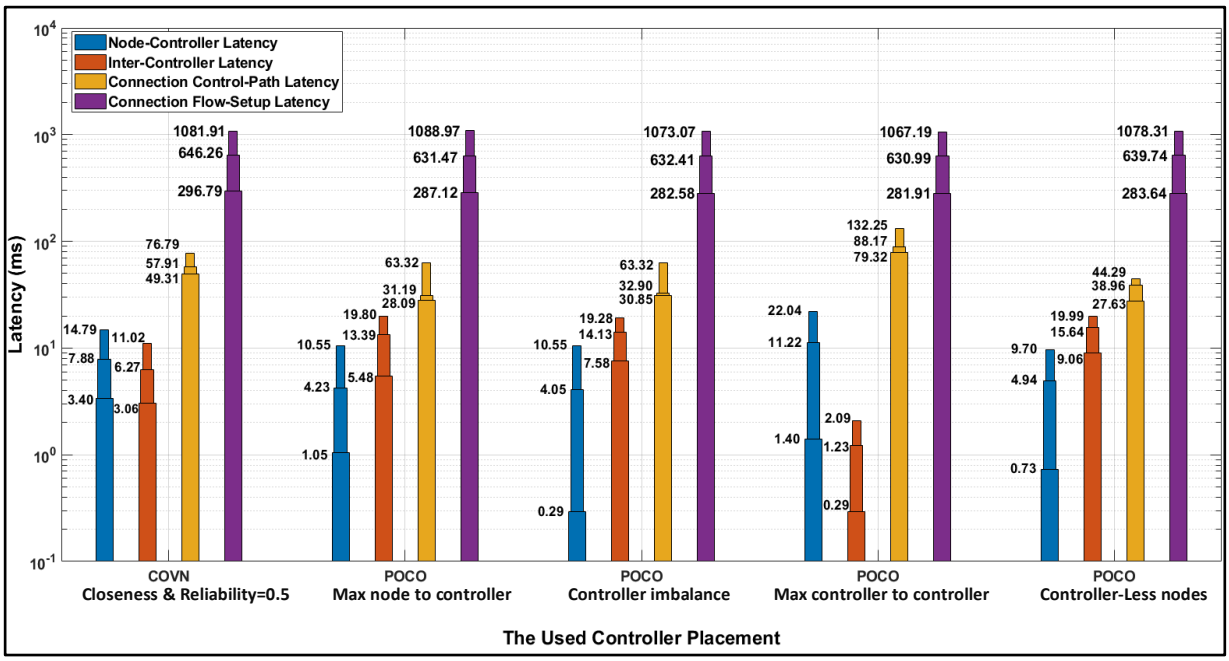


Figure 6.46: The comparison of latency metrics between the COVN and POCO placements on the real distances of Internet2 for the failure-free of the controller.

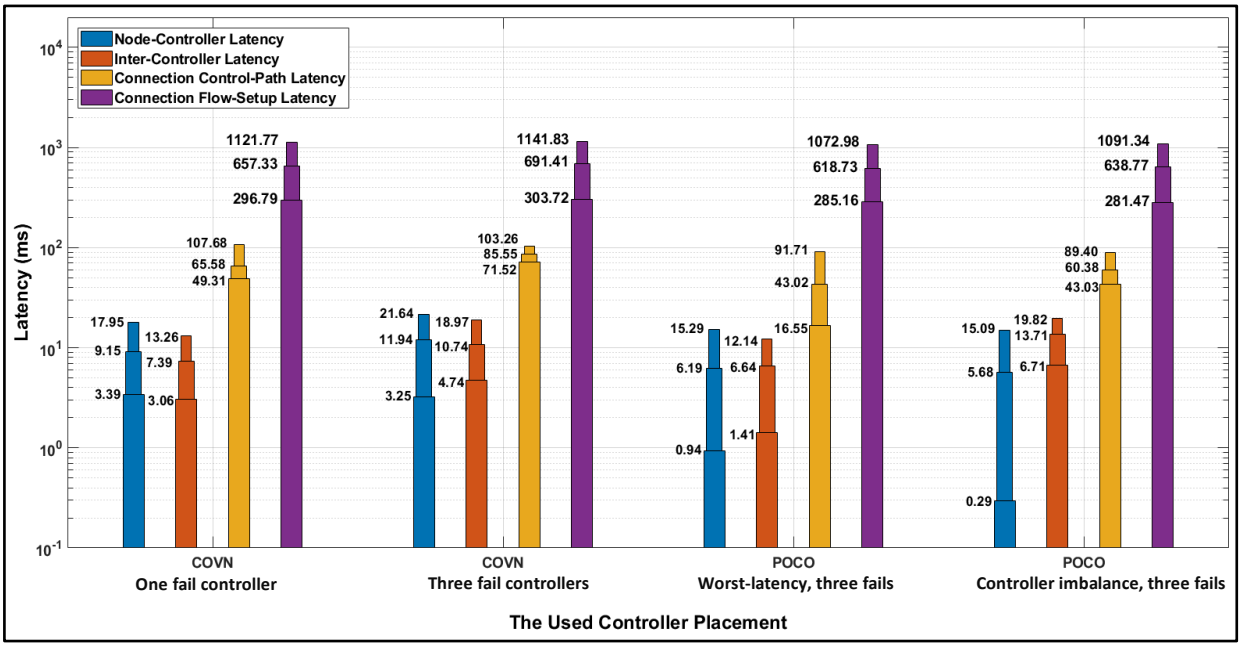


Figure 6.47: The comparison of latency metrics between the COVN and POCO placements on the real distances of Internet2 for the different failures of the controller.

Figure 6.46 shows a comparison of five different groups of the latency metrics, without considering the controller failure cases (failure-free). The first group of the latency metrics is produced by the COVN placement, while the other four groups are created by

POCO algorithm for different objectives. On the one hand, there is a similarity in the behaviour of the latency metrics of the first group (COVN placement latencies) and the fourth group (Max controller to controller placement latencies), although, their values are different. On the other hand, there is a dissimilarity in the behaviour and the value between the COVN placement latencies and the POCO placement latencies of the second, third and fifth groups. The COVN placement latencies have the node-controller latency larger than the inter-controller latency, but they are not far from each other (balancing between them). The two latencies of the fourth group behave similarly, but the difference between them is larger (unbalancing between them). In contrast, the other three latency groups of POCO placement, have the node-controller latency smaller than inter-controller latency. The final analysis of the latency metrics in Figure 6.46 demonstrates that:

1. Reducing the node-control latency and ignoring the optimisation of the inter-controller latency (second group), generates the ***larger connection flow-setup*** latency.
2. Placing the controllers on any objective that keep the node-controller latency smaller than inter-controller latency (third and fifth groups), produces good connection flow-setup latency. However, the high inter-controller latency degrades the controller convergence and the resources utilisation.
3. Reducing the inter-controller latency and ignoring the optimisation of the node-controller latency, produces the smallest connection flow-setup latency (see the fourth group).
4. COVN placement kept balancing between the node-controller latency and inter-controller latency to obtain a good connection flow-setup latency (***higher than the lowest latency in the fourth group by 14 ms***). Also, to gain the central location (optimal closeness), this serves the controller placement of the dynamic VN which is not considered in the POCO placement algorithm.

After that, Figure 6.47 exposes the comparisons between the group of four tests of the latency metrics, with considering the occurrence of the controller-failure. The first two groups of latency metrics are calculated for the COVN placement, when one and three controllers are failed. While, the last two groups of latency metrics introduce the POCO placement latencies. The third group of latency metrics aims to optimise the worst-case

latency between the node and its controller. Whereas, the fourth group intends to optimise the load balance between controllers, when the controller-failure probability is up to the three controllers. Figure 6.47 demonstrates better latency metrics of POCO algorithm than the COVN placement algorithm, in the failure case. However, the COVN placement keeps a balanced load of the controller by placing the responsibility of the failed controller to a backup controller. While, the POCO algorithm degrades the load balance by re-distributing the load of the failed controller on the other active controllers. Also, it is noteworthy that the connection flow-setup latency of the COVN placement algorithm is higher than the latency of POCO algorithm by **69 ms** in the worst case (the difference between the second and third groups). The difference is large because of the large geographical expansion of the Internet2 topology, which considers the worst-case of implementation for COVN placement algorithm as will be demonstrated in the next section.

B) Latency metrics for the minimised length of links of Internet2 topology

The two figures below (Figures 6.48 and 6.49) present the same previous comparisons of latency metrics, but for the minimised length of links of Internet2 topology. The purpose of this, was to prove that the COVN placement algorithm could present better results of latency for this network when it had a smaller expansion. This supports the validity of the facts that the COVN placement performs better for networks which have restricted geographical expansion.

One circumstance shows that, the node-controller, inter-controller and connection control-path latency metrics in Figures 6.48 and 6.49below, indicate a similarity in the attitude of those in Figures 6.46 and 6.47 above. The network of the minimised length of links produces smaller values of the mentioned three latency metrics, but they are directly proportional with their peers in the network of the real length of links.

Another circumstance shows that, the connection flow-setup latency changes its behaviour in the second case (minimised length of links). Analyzing the results in Figure 6.48 revealed that:

1. Reducing the node-control latency to be smaller than the inter-controller latency, creates a good connection flow-setup latency (see group 2, 3 and 5).
2. Reducing the inter-controller latency and ignoring the optimisation of the node-controller latency, produces the heights (worst) connection flow-setup latency (see group 4).

3. COVN placement kept balancing between the node-control latency and inter-controller latency to obtain a good connection flow-setup latency (***higher than the lowest latency in the third group by 0.45 ms***).

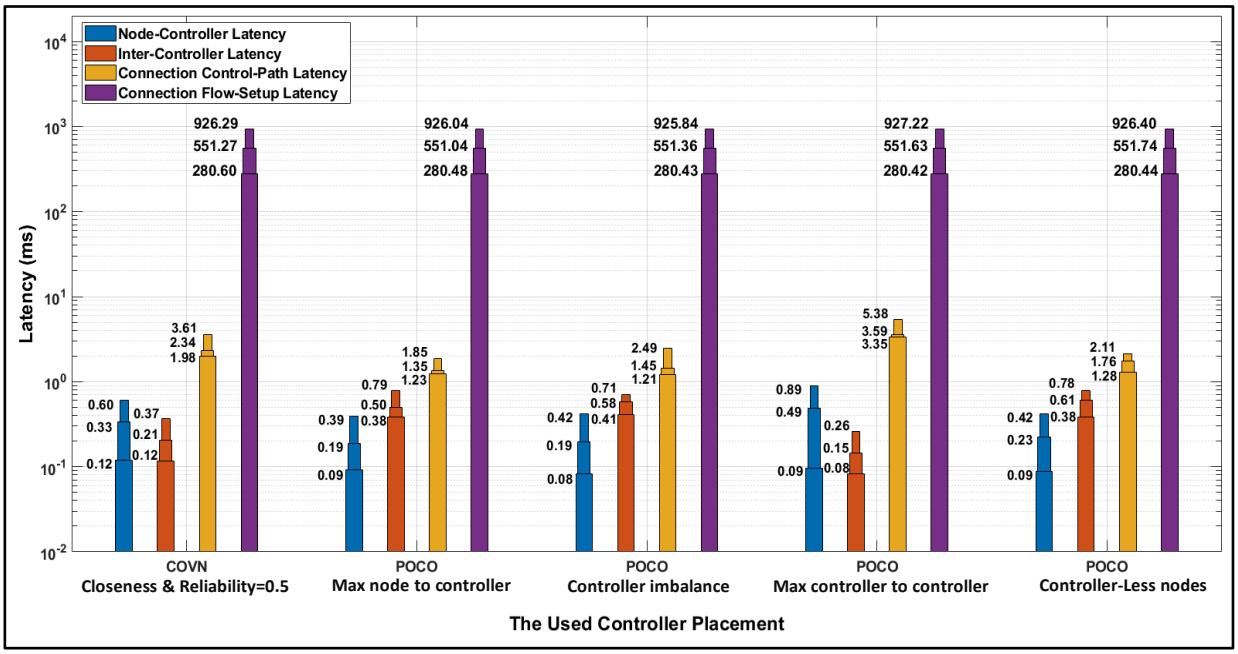


Figure 6.48: The comparison of latency metrics between the COVN and POCO placements on the minimised distances of Internet2 for the failure-free of the controller.

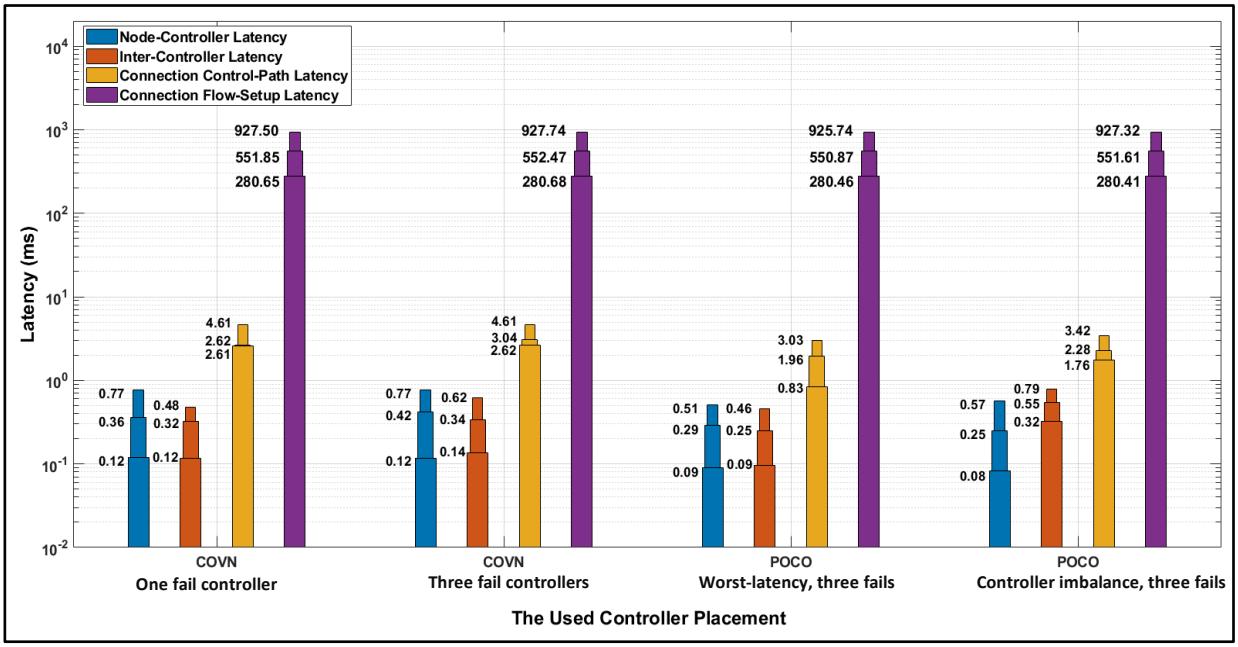


Figure 6.49: The comparison of latency metrics between the COVN and POCO placements on the minimised distances of Internet2 for the different failures of the controller.

Figure 6.49 shows that connection flow-setup latencies for COVN placement are higher than those of POCO placement, but this time the difference in worst case is only 1.53 ms (the difference between the second and third groups).

The findings from the comparison of the two cases A and B disclose that:

1) The balancing between the node-controller latency and inter-controller latency obtains an acceptable connection flow-setup latency for both cases (A and B).

2) The performance of the COVN placement algorithm is better for the network with smaller lengths of links.

The third evaluation comparison examines the reliability metrics of the COVN and POCO placement algorithm as shown in Figures 6.50 and 6.51 below.

Figure 6.50 presents the five group of tests of COVN and POCO placement, without considering the controller-failure occurrence. It demonstrates that the controller-connectivity ratio of the COVN placement algorithm is higher (better) than the other four of the POCO algorithm. However, the length of the control-path (Min-Hops to Farthest node) is higher than others of POCO algorithm by one or two hops.

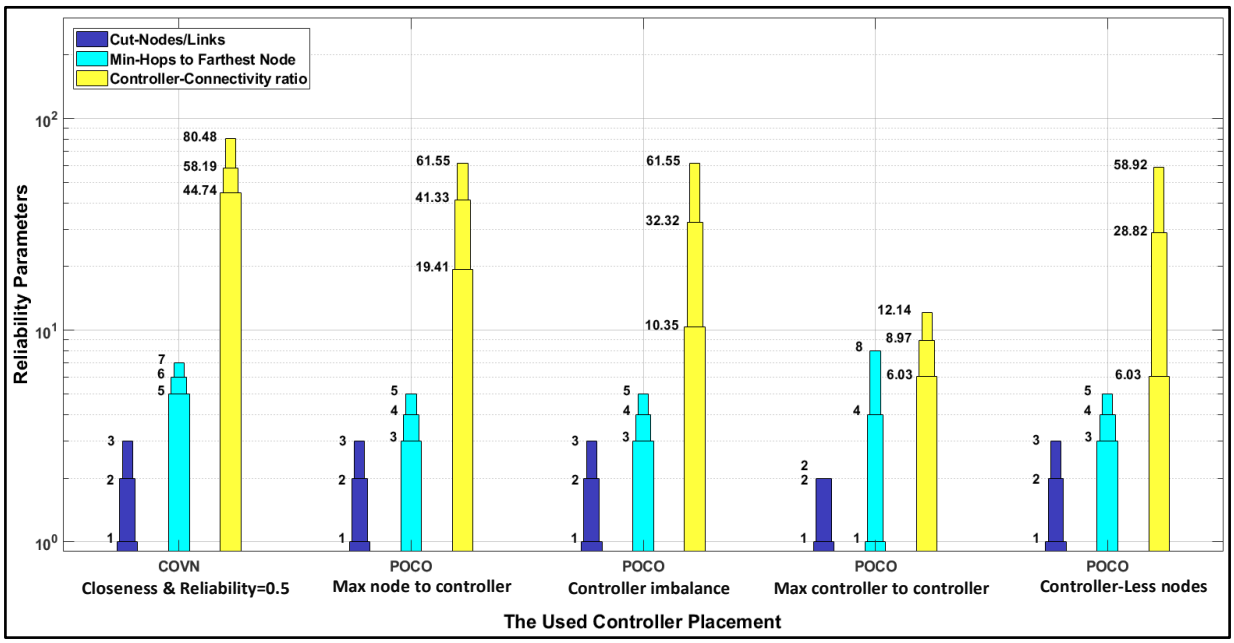


Figure 6.50: The comparison of reliability metrics between the COVN and POCO placements on the real distances of Internet2 for the failure-free of the controller.

Figure 6.51 presents the four group of two controller placement algorithm tests, considering the probability of the controller failure occurrence up to three failures. The figure illustrates that the controller-connectivity ratio is better for the COVN placement than POCO placement. Also, it expresses that the control-path is longer for the COVN placement than POCO placement.

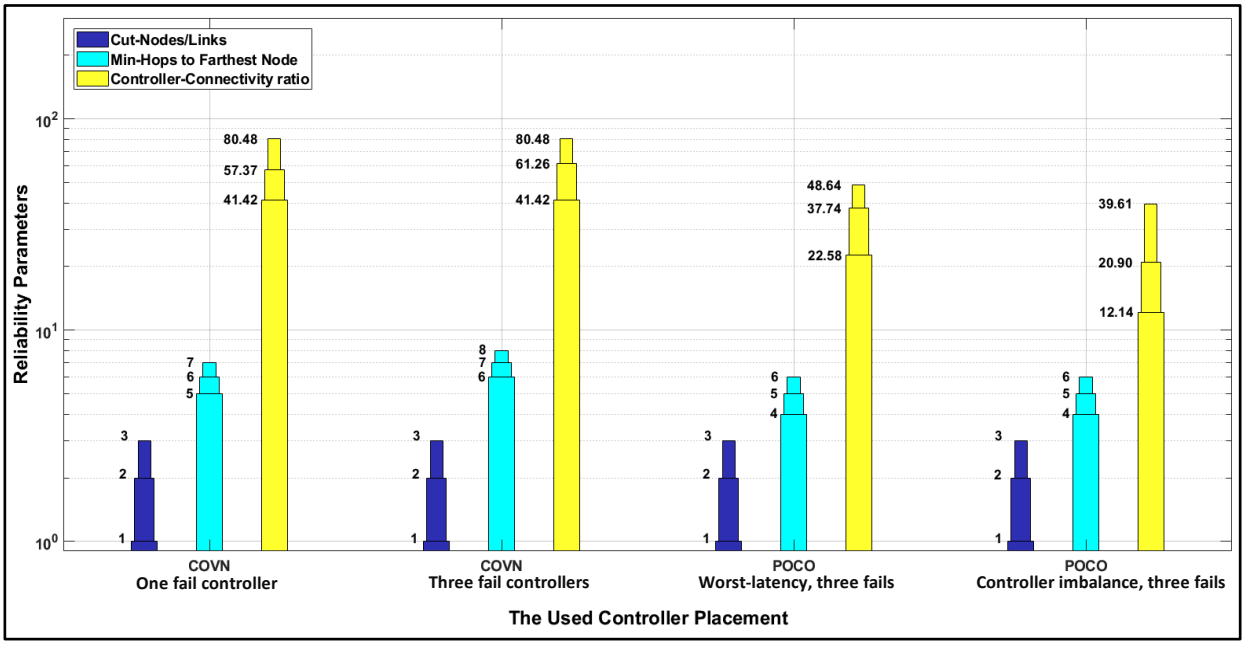


Figure 6.51: The comparison of reliability metrics between the COVN and POCO placements on the real distances of Internet2 for the different failures of the controller.

In general, the COVN placement produces optimal reliability for SD-WAN which is better than POCO algorithm.

6.6. Summary and Conclusion

This chapter shows the used topology, the reasons for using them and their characteristics. Then it explains the objectives, the test strategy and the findings for the six categories of the tests of the COVN simulator. Subsequently, it validates the results of the COVN simulator by applying some similar tests on Mininet emulator and comparing the results for both. Finally, the chapter evaluates the performance of the COVN placement algorithm by comparing its load balancing, latency and reliability metrics with the similar metrics which are calculated depending on the controller placement of the POCO algorithm. The objectives and the findings of all the above sections are summarised in Table 6.16 below.

Testbed	Test Objectives	Findings
COVN simulator	**Basic metrics:** Validate the basic assumptions of the COVN placement.	1. ***The COVN algorithm keeps a small and restricted inter-controller latency for optimising: the connection flow-setup latency; the controller's convergence and resources utilisation.*** 2. ***The connection control latency has a small effect on connection flow-setup latency.*** 3. ***The controller nodes which are selected by COVN algorithm have the best reliability parameters in comparison to other nodes of SD-WAN.*** 4. ***Recovering the failed controller does not lead to the recalculating of the controller placement or clustering the topology to occur.***
	Closeness and Reliability weights: Exploring a method for determining the optimal weights of the closeness and the reliability.	5. ***The optimal weights of closeness and reliability would always depend on the placement requirements which serve a specific network and its services. While, the decision made in this thesis tries to balance between the latency and the reliability metrics.*** 6. ***The networks where the majority of its links length are less than 100 Km produce optimal controller placement according to the latency when using the CONV algorithm. However, the resulted placement when applying COVN algorithm on Internt2 resulted in acceptable latency metrics.*** 7. ***Using the controller placement to improve the network reliability is more effective than using it to improve its latency.***
	Clustering: Investigating: The ability to create balanced clusters; And the effect of increasing the number of clusters on the results.	8. ***The COVN placement algorithm can efficiently create balanced clusters.*** 9. ***The node-controller and connection control-path latencies decrease with increasing the number of clusters, also the inter-controller latency rises with the increasing number of clusters.*** 10. ***The connection flow-setup latency is affected by the difference between the connection-control path and inter-controller latency.*** 11. ***The reliability parameters are almost stable when increasing the number of clusters.***

	Controller-failure: Investigating: The ability of the quick recovery of controller-failure; and the effect of this process on the metrics.	12. ***The COVN algorithm can perform fast recovery for controller failure.*** 13. ***The latency metrics receive a tiny increment in a microsecond, consequently connection flow-setup latency increases by only some milliseconds.*** 14. ***The reliability metrics are degraded by an acceptable value, which does not demolish the SD-WAN reliability.***
	Changes of the VNs: Showing the reaction of the COVN placement algorithm to the changes of the VNs and proving the efficiency of this reaction.	15. ***The COVN placement algorithm can efficiently keep the balance of the clusters of the VN.*** 16. ***The changes to the VN produces unavoidable changes in latency and reliability metrics. However, the COVN placement algorithm is designed to limit these changes to the smallest values needed to keep sufficient performance.***
	Network size: Investigating the limitation of the following: the number of the placed controller; the increment in the connection flow-setup latency; and the execution time of the algorithm.	17. ***The COVN placement algorithm can efficiently increase the number of the placed physical and virtual controllers with the increasing the size of the network.*** 18. ***The latency of VN raises with the increasing size of the network. However, it acquires a higher increment with the network of the wider expansion like Us-carrier and Kentucky.*** 19. ***The COVN algorithm keeps a low execution time even when increasing the size of the network and the number of the placed controllers.***
Mininet Emulator	Proving the validity of the simulation results, namely the connection flow-setup latency.	20. ***The comparison demonstrates a matching between the connection flow-setup latency of the COVN simulator and its peer in Mininet emulator. This proves the validity of the computations and results of COVN simulator.***
Comparison with results of POCO algorithm	Evaluating the controller placements of the COVN placement algorithm.	21. ***The COVN placement produces a better balancing in the number of nodes than the POCO placement.*** 22. ***The COVN placement also produces the load balancing, which depends on the load of the network nodes (requests of the flow-rule per seconds).***

		23. ***The COVN produces an acceptable connection flow-setup latency by considering the optimisation of both the node-controller and inter-controller latency.*** 24. ***The difference between the connection flow-setup latency of the COVN and POCO placement is very small and can be ignored (less than 1 ms), when the length of network links is less or equal to 20 km.*** 25. ***The COVN placement resulted in more reliable controller placements for SD-WAN than the POCO placement.***

Table 6.16: Summary of the findings for the tests of all the sections presented in chapter seven.

The findings in the above table reflect the success of achieving the objectives that are required for implementing the COVN placement algorithm over SD-WAN with multiple virtual networks. Also, they demonstrate that the COVN placement algorithm could effectively place the controller for any topology, because it produces better load balancing and reliability metrics and acceptable latency metrics. However, its optimal performance could be gained when applying it on restricted expansion topology, which has a dynamic load variation. Finally, it is noteworthy that this algorithm provides a new method for placing the virtual controllers and recovering the controller-failure, which are not presented in the previous placement algorithms.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK

This thesis acknowledges the importance of deploying the virtual slices of SD-WAN to serve the forthcoming networks such as 5G cellular network and smart city network.

The deployment of this new model motivates the author to construct a novel controller placement algorithm, called COVN placement, to locate the controllers of the SD-WAN with multiple virtual networks. Although this issue has been dealt with previously, only the physical SD-WAN was considered.

The solution is based on a good understanding of the released standard of SDN and NFV, which are published by verified organisations such as ONF. In addition, the solution considers the intensive search of related works as demonstrated in chapter two. The findings of this chapter lead to the following decisions (see Table 2.6), which are considered in designing the algorithm: 1) Minimising the inter-controller path optimises the end-to-end latency and network utilization (Zhang, Bianco and Giaccone, 2016)(He *et al.*, 2017); 2) The reliability of controller placement could be improved without creating unsatisfactory latency (Hu *et al.*, 2014); 3) The executive search method of the high computation complexity (Sahoo *et al.*, 2017) could be replaced with a new method which uses the hop count to reduce the computational complexity. However, the third point depends on the delay at the hop and whether this delay could be used instead of the propagation latency of the network links.

The literature review provides some guidelines but does not offer the complete solution. Therefore, the novel SFOP routing algorithm is built to examine whether it is possible to optimise the network throughput by using the path of feasible bandwidth. The results show that, the control packets are small and do not degrade the network bandwidth (see Section 3.2).

After that, two measurement techniques are developed on physical equipment, to investigate the value of the link and the hop delays in SD-WAN from practical tests. The investigation leads to define two new latency metrics, which are the connection control-path latency and connection flow-setup latency (see Section 3.3.5). These two latencies

help to identify the ratio of effecting the controller placement on the end-to-end latency. Also, it proves that the hop latency could be used instead of the link propagation latency for small and medium expansion networks (links less or equal to 20 Km).

The design stage comes later to produce the COVN placement algorithm, which places the controller of every VN independently with considering the status of other VNs. COVN placement maintains low complexity by applying the new graph theorem using hop count to place the controllers (see Section 4.7.3) and novel peripheral clustering algorithm to distribute the nodes on the controllers. This peripheral clustering is designed in this research too (see Section 4.8). COVN placement optimises the latency and reliability of virtual SD-WANs according to the desired weight. Also, it optimises the load balancing and the utilisation of the controllers.

The implementation is performed by the COVN simulator, produced in this research, and also by the developed Mininet emulator. Furthermore, the performance of COVN placement is validated by comparing its resulting performance metrics with the famous POCO placement algorithm. The six different categories of tests, which are applied on real network topology and the comparisons with POCO results prove that COVN placement algorithm can efficiently place the controllers of this model (see Table 6.16). Finally, COVN placement provides recovery for the controller-failure, with keeping a minimum number of backup controllers.

Inventing COVN placement algorithm significantly contributes to optimising the control plane of multiple virtual SD-WANs in terms of the following objectives: reliability, latency, load balance and resources utilisation. Examining the objectives, designing COVN placement and testing its abilities, led to several contributions, which are demonstrated as follows.

7.1. Novel SFOP Routing Algorithm

This routing algorithm exploits the new abilities of the OpenFlow protocol and SDN controller, which are the fine granularity statistics of all flows. Using these statistics allows the routing algorithm to choose the shortest-path which has the feasible bandwidth for the served flow. SFOP routing optimises the usage of the network bandwidth without additional cost for collecting statistics but slightly increases the computation complexity. This algorithm along with traditional routing algorithms are

employed to investigate the possibility of adding specific routes between the controller and its switches to optimise the bandwidth utilisation. However, the results reveal that the control packets are small and do not degrade the bandwidth (see Section 3.2).

7.2. Two Latency Measurement Technique

Measuring the latency of SDN on the physical and emulated testbeds requires finding a measurement technique which is convenient for both and provides accurate results. Using the controller to measure the latency of data and control flow is avoided in this research. This is because, it was proved by previous studies (Shuo Wang *et al.*, 2017)(Atary and Bremler-Barr, 2016) as well as the present research that the latency of the control-plane continuously changes and negatively affects the measurement accuracy. Therefore, the hosts in data-plane and the traditional ping tool were used along with a new mechanism that eliminates the additional latency in the receiver side. Removing the latency of the second side makes the latency measurement more accurate and then it is used to measure the latency of data and control packets (see Sections 3.3 and 3.4). The measurement produces two new latency metrics and proves that hop latency could be used to find the shortest path for medium distance networks. Also, it contributes to formulating the latency equation (see Section 3.3.5).

7.3. Novel Peripheral Clustering Algorithm

Before accomplishing the COVN placement, the study requires the researcher to identify a clustering algorithm, which: has a low computation complexity; creates good load balancing; and produces converged clusters of connected nodes. Combining these requirements motivates the author to design the peripheral clustering algorithm, which exploits several aspects. First, using the geographical distance instead of the length of links to minimise the computation of determining the connected converged nodes. Second, balancing the clusters according to the variation between them in two different ways to produce equal loads in low computation complexity. Third, representing the graph as a sparse matrix and forming the cluster as a single object, which reduces the computation of the search operation. All the above steps reduce the computational complexity and produce a clustering algorithm which uses the programming intelligence instead of the iteration to create balanced clusters.

7.4. Novel COVN Placement

COVN placement algorithm completely differs from all previous placement algorithms, because it deals with multiple virtual networks running over single physical SD-WAN. While, the previous works only place the controllers of a single physical SD-WAN. First, COVN placement needs to determine the optimal nodes in the complete SD-WAN without partitioning it, to place the physical controllers. This is because, these physical controllers should serve all the virtual controllers of all the VNs, which have various topologies, loads and number of controllers. Then, the controller placement of every VN should perform independently according to its topology and load. This requires clustering the VN according to the maximum load of every virtual controller and then assigning it to every cluster. The positions of the virtual controllers are selected among those of the physical controllers in order to produce minimum equal latency between the controller and its switches for all clusters.

The placement of the physical controller intends to maximise the reliability and minimise the latency. These two objectives are combined according to the selected weight which is determined according to the SD-WAN requirements. The placement of the virtual controllers concerns the load balancing and the recourses utilisation because the most alternation occurs in the VNs.

The results of COVN placement algorithm show that: 1) It always produces a reliable controller placement; 2) It always maintains the small path between controllers, which optimises the connection flow-setup latency, convergence latency of the controllers and resources utilisation; 3) It gains better latency optimisation for the network of small and medium expansion, and also for the network which has an approximately similar length of links. 4) It keeps the load balancing when changing the number of clusters or the topology of the VN; 5) It provides fast recovery for the controller-failure with small degradation in latency and reliability; and 6) It performs the controller placement for a large size of network and a large number of controllers in small computation time (several minutes).

Finally, comparing COVN placement with POCO algorithm revealed that COVN placement produces better reliability and similar latency. In addition, it surpasses POCO by having the ability to balance the load according to the number of requests per second. It also overrides POCO by providing the solution of recovering the controller failure.

7.5. COVN TestBeds

This research constructs two experimental platforms to perform the tests. It first programs a new testbed, which is the COVN simulator. This testbed applies the two parts of COVN placement algorithm. It also generates VN topology, its load matrix and its bandwidth matrix. Then, it calculates the latency and reliability statistics for the placement implementation.

Moreover, the researcher developed the Mininet emulator to be able to implement the controller placement.

The resulting contributions can be used to improve the control plane of this new model of the network and also to evaluate its performance and functionality. This research contributes to serving the implementation of the future networks such as the 5G cellular system and smart city network.

7.6. Future Work

The contributions of this research could be improved further, for instance:

1. COVN placement algorithm could be extended to improve other objectives, such as minimising power consumption by reducing the number of active links, when eliminating the redundant paths.

2. Developing COVN placement algorithm to be integrated with the management applications of 5G cellular system. It can play a big role in optimising its backbone network. This is because the backbone network serves a large number of small cells, which have a dynamic load and connectivity.

3. Developing COVN placement algorithm to perform auto selection for the weight of closeness against the weight of the reliability by determining the worst threshold for latency or reliability of SD-WAN.

4. Investigating the most convenient techniques for converging the controllers, which are placed by the COVN placement algorithm and finding out the extent to which it is possible to minimise the convergence latency.

REFERENCES

Abdelaziz, A., Fong, A.T., Gani, A., Garba, U., Khan, S., Akhunzada, A., Talebian, H., Choo, K.-K.R. (2017) Distributed controller clustering in software defined networks. *PLOS ONE*. **12**(4), e0174715.

Abe, J.O., Mantar, H.A., Yayimli, A.G. (2015) k -Maximally Disjoint Path Routing Algorithms for SDN. In *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, pp. 499–508.

ACG RESEARCH (2014) Business Case for Cisco SDN for the WAN. *www.acgresearch.net.* [online]. Available from: https://www.cisco.com/c/dam/en/us/products/collateral /routers/wan-automation-engine/business-case-for-cisco-sdn-for-the-wan.pdf [Accessed July 13, 2017].

van Adrichem, N.L.M., Doerr, C., Kuipers, F.A. (2014) OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, pp. 1–8.

Adrichem, V., M., N.L., Doerr, C., Kuipers, F.A. (2014) OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, pp. 1–8.

Agarwal, K., Rozner, E., Dixon, C., Carter, J. (2014) SDN traceroute: Tracing SDN Forwarding without Changing Network Behavior. *Proceedings of the third workshop on Hot topics in software defined networking. ACM*, 145–150.

Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P. (1998) Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD Record*. **27**(2), 94–105.

Akyildiz, I.F., Wang, P., Lin, S.C. (2015) SoftAir: A software defined networking architecture for 5G wireless systems. *Computer Networks*. **85**, 1–18.

Al-Jawad, A., Trestian, R., Shah, P., Gemikonakli, O. (2015) BaProbSDN: A probabilistic-based QoS routing mechanism for Software Defined Networks. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, pp. 1–5.

Al-Sadi, A., Al-Sherbaz, A., Xue, J., Turner, S. (2016) Management of Distributed Software-Defined Networks in Smart Cities. In *School of Science and Technology Annual Research Conference, The University of Northampton*. Northampton-UK: The University of Northampton, pp. 1–6.

Alasadi, E., Al-raweshidy, H. (2018) OLC : Open-Level Control plane architecture for providing better scalability in an SDN network. *IEEE Access*. **PP**(c), 1.

Aloul, F., Rawi, B., Aboelaze, M. (2006) Identifying the Shortest Path in Large Networks using Boolean Satisfiability. In *2006 3rd International Conference on Electrical and Electronics Engineering*. IEEE, pp. 1–4.

Altukhov, V., Chemeritskiy, E. (2014) On real-time delay monitoring in software-defined networks. In *2014 First International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)*. IEEE, pp. 1–6.

Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J., Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J. (1999) OPTICS: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data - SIGMOD '99*. New York, New York, USA: ACM Press, pp. 49–60.

Aoki, H., Nagano, J., Shinomiya, N. (2015) Network Partitioning Problem to Reduce Shared Information in OpenFlow Networks with Multiple Controllers. In *ICN 2015 : The Fourteenth International Conference on Networks*. Barcelona, Spain: IARIA, pp. 250–255

Aoki, H., Shinomiya, N. (2016) Controller Placement Problem to Enhance Performance in Multi-domain SDN Networks. In *CICN 2016 : The Fifteenth International Conference on Networks*. Tokyo, Japan, pp. 108–110.

Aoki, H., Shinomiya, N. (2015) Network Partitioning Problem for Effective Management of Multi-domain SDN Networks. *International Journal on Advances in Networks and Services*. **8**(3–4), 62–77.

Atary, A., Bremler-Barr, A. (2016) Efficient Round-Trip Time monitoring in OpenFlow networks. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. IEEE, pp. 1–9.

Auroux, S., Draxler, M., Morelli, A., Mancuso, V. (2014) CROWD Dynamic Network Reconfiguration in Wireless DenseNets. In *European Conference on Networks and Communications (EuCNC'14)*. pp. 1–6.

Auroux, S., Draxler, M., Morelli, A., Mancuso, V. (2015) Dynamic network reconfiguration in wireless DenseNets with the CROWD SDN architecture. In *2015 European Conference on Networks and Communications (EuCNC)*. IEEE, pp. 144–148.

Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., Xiao, X. (2002) Overview and Principles of Internet Traffic Engineering. *Network Working Group*. [online]. Available from: https://tools.ietf.org/html/rfc3272#page-4 [Accessed July 13, 2018].

Bannour, F., Souihi, S., Mellouk, A. (2017) Scalability and reliability aware SDN controller placement strategies. In *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, pp. 1–4.

Barbehenn, M. (1998) A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*. **47**(2), 263.

Bari, M.F., Roy, A.R., Chowdhury, S.R., Zhang, Q., Zhani, M.F., Ahmed, R., Boutaba, R. (2013) Dynamic Controller Provisioning in Software Defined Networks. In *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*. IEEE, pp. 18–25.

Beckhoff Automation GmbH & Co. KG, G. (2017) *Infrastructure for EtherCAT/Ethernet Technical recommendations and notes for design, implementation and testing Table of contents*.

Bellaachia, A., Portnoy, D., Chen, Y., Elkahloun, A.G., Cgb, N.I.H.N. (2002) E-CAST : A Data Mining Algorithm For Gene Expression Data. In *Proceedings of the 2nd International Conference on Data Mining in Bioinformatics*. pp. 49–54.

Berkhin, P. (2006) A Survey of Clustering Data Mining Techniques. In *Grouping Multidimensional Data*. Berlin/Heidelberg: Springer-Verlag, pp. 25–71.

Berrar, D.P., Dubitzky, W., Granzow, M. (2003) *A practical approach to microarray data analysis*. New York: Kluwer Academic Publishers.

Bobrovs, V., Spolitis, S., Ivanovs, G. (2014) Latency causes and reduction in optical metro networks. *In Optical Metro Networks and Short-Haul Systems VI*. **9008**, 90080C. International Society for Optics and Photo.

Borcoci, E., Badea, R., Obreja, S.G., Vochin, M. (2015) On Multi-controller Placement Optimization in Software Defined Networking - based WANs. In *ICN 2015, the Fourteenth International Conferences on Networks*. pp. 261–266.

Brosi, J.-M. (2008) *Slow-Light Photonic Crystal Devices for High-Speed Optical Signal Processing*. Univ.-Verlag Karlsruhe.

Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., Schulz, C. (2015) Recent Advances in Graph Partitioning. *Lecture Notes in Computer Science*. **9220**, 117–158.

Castro, J., Georgiopoulos, M., Demara, R., Gonzalez, A. (2005) *Data-partitioning using the Hilbert space filling curves: Effect on the speed of convergence of Fuzzy ARTMAP for large database problems*.

Cello, M., Xu, Y., Walid, A., Wilfong, G., Chao, H.J., Marchese, M. (2017) BalCon: A Distributed Elastic SDN Control via Efficient Switch Migration. In *2017 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, pp. 40–50.

Chiosi, M., Clarke, D., Willis, P., Reid, A., Feger, J., Bugenhagen, M., Khan, W., Fargano, M., Cui, C., Deng, H., Telekom, D., Michel, U. (2012) Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action. In *SDN and OpenFlow World Congress*. pp. 22–24.

Cisco (2016) Cisco SFP Modules for Gigabit Ethernet Applications Cost-effective Small Form-factor Pluggable (SFP) transceivers for Gigabit Ethernet applications. *Cisco Public Information*. **C78-366584**, 1–9.

Cox, J.H., Chung, J., Donovan, S., Ivey, J., Clark, R.J., Riley, G., Owen, H.L. (2017) Advancing Software-Defined Networks: A Survey. *IEEE Access*. **5**, 25487–25526.

Danielis, P., Altmann, V., Skodzik, J., Schweissguth, E.B., Golatowski, F., Timmermann, D. (2015) Emulation of SDN-supported automation networks. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, pp. 1–8.

Dempster, A.P., Laird, N.M., Rubin, D.B. (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*. **39**(1), 1–38.

Dhaval N. Shah, Virupaksh Honnur, Dalen, D. Bosteder (2002) System and method for measuring round trip times in a network using a TCP packet.

Dixit, A., Hao, F., Mukherjee, S., Lakshman, T.V., Kompella, R., Dixit, A., Hao, F., Mukherjee, S., Lakshman, T.V., Kompella, R. (2013) Towards an elastic distributed SDN controller. *ACM SIGCOMM Computer Communication Review*. **43**(4), 7–12.

Donath, W.E., Hoffman, A.J. (1972) Algorithms for partitioning graphs and computer logic based on eigenvectors of connection matrices. In *IBM Technical Disclosure Bulletin, vol. 15, no. 3,pp. 938–944*.

Durgin, N., Mai, Y., Randwyk, J. Van (2005) NetState : A Network Version Tracking System. In *USENIX Annual Technical Conference, FREENIX Track*. pp. 19–127.

Egilmez, H.E., Civanlar, S., Tekalp, A.M. (2013) An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks. *IEEE Transactions on Multimedia*. **15**(3), 710–715.

Egilmez, H.E., Dane, S.T., Bagci, K.T., Tekalp, A.M. (2012) OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, 1–8.

Egilmez, H.E., Gorkemli, B., Tekalp, A.M., Civanlar, S. (2011) Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing. In *2011 18th IEEE International Conference on Image Processing*. IEEE, pp. 2241–2244.

Everitt, B.S., Landau, S., Leese, M., Stahl, D. (2011) *Cluster Analysis*. Chichester, UK: John Wiley & Sons, Ltd.

Fekih Ahmed, M., Talhi, C., Cheriet, M. (2015) Towards flexible, scalable and autonomic virtual tenant slices. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, pp. 720–726.

Fiedler, M. (1975) A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*. **25**(4), 619–633.

Ford, L.R., Fulkerson, D.R. (2009) Maximal Flow Through a Network. In *Classic Papers in Combinatorics*. Boston, MA: Birkhäuser Boston, pp. 243–248.

Foundation, O.N. (2015) Software-Defined Networking (SDN) Definition. *Open Networking Foundation*. [online]. Available from: https://www.opennetworking.org/sdn-resources/sdn-definition [Accessed December 2, 2018].

George, A., Liu, J.W.H. (1984) Computer solution of large sparse positive definite systems. *Society for Industrial and Applied Mathematics*. **26**(2), 289–291.

Goil, S., Nagesh, H., Choudhary, A. (1999) MAFIA: Efficient and scalable subspace clustering for very large data sets. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 443–452.

Guha, S., Rastogi, R., Shim, K. (2000) Rock: A robust clustering algorithm for categorical attributes. *Information Systems*. **25**(5), 345–366.

Guha, S., Rastogi, R., Shim, K., Guha, S., Rastogi, R., Shim, K. (1998) CURE: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data - SIGMOD '98*. New York, New York, USA: ACM Press, pp. 73–84.

Guo, M., Bhattacharya, P. (2013) Controller Placement for Improving Resilience of Software-Defined Networks. In *2013 Fourth International Conference on Networking and Distributed Computing*. IEEE, pp. 23–27.

Gustaf Jan Gunnar Nilstadius (2016) *Software defined networks with high availability*. D. of T. E. Czech technical university, ed. Czech technical university, Department of Telecommunication Engineering.

Han, J., Kamber, M., Tung, A.K.H. (2001) Spatial Clustering Methods in Data Mining: A Survey. *Geographic Data mining and knowledge discovery*. **2**, 188–217.

Han, L., Li, Z., Liu, W., Dai, K., Qu, W. (2016) Minimum Control Latency of SDN Controller Placement. In *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, pp. 2175–2180.

He, M., Basta, A., Blenk, A., Kellerer, W. (2017) Modeling flow setup time for controller placement in SDN: Evaluation for dynamic flows. In *2017 IEEE International Conference on Communications (ICC)*. IEEE, pp. 1–7.

Hegde, S., Ajayghosh, R., Koolagudi, S.G., Bhattacharya, S. (2017) Dynamic controller placement in edge-core software defined networks. In *TENCON 2017 - 2017 IEEE Region 10 Conference*. IEEE, pp. 3153–3158.

Heller, B. (2013) *Reproducible Network Research With High-Fidelity Emulation*. (Doctoral dissertation, Stanford University).

Heller, B., Sherwood, R., McKeown, N. (2012) The controller placement problem. In ACM, ed. *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*. New York, New York, USA: ACM Press, pp. 7–12.

Hendrickson, B., Leland, R. (1995) An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations. *SIAM Journal on Scientific Computing*. **16**(2), 452–469.

Hinneburg, A., Keim, D.A. (1998) An Efficient Approach to Clustering in Large Multimedia Databases with Noise. *Proceedings of the 4th ACM SIGKDD Knowledge Discovery and Data Mining AAAI Press, Menlo Park*. **5865**(c), 58–65.

Hock, D., Gebert, S., Hartmann, M., Zinner, T., Tran-Gia, P. (2014) POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, pp. 1–2.

Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., Tran-Gia, P. (2013) Pareto-optimal resilient controller placement in SDN-based core networks. In *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*. IEEE, pp. 1–9.

Hock, D., Hartmann, M., Gebert, S., Zinner, T., Tran-Gia, P. (2014) POCO-PLC: Enabling dynamic pareto-optimal resilient controller placement in SDN networks. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, pp. 115–116.

Hollinghurst, J., Ganesh, A., Bauge, T. (2016) Controller placement methods analysis. In *2016 6th International Conference on Information Communication and Management (ICICM)*. IEEE, pp. 239–244.

Hu Bo, Wu Youke, Wang Chuan'an, Wang Ying (2016) The controller placement problem for software-defined networks. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, pp. 2435–2439.

Hu, Y., Luo, T., Beaulieu, N.C., Deng, C. (2017) The Energy-Aware Controller Placement Problem in Software Defined Networks. *IEEE Communications Letters*. **21**(4), 741–744.

Hu, Y., Wang, W., Gong, X., Que, X., Cheng, S. (2014) On reliability-optimized controller placement for Software-Defined Networks. *China Communications*. **11**(2), 38–54.

Hu, Y., Wendong, W., Gong, X., Que, X., Shiduan, C. (2013) Reliability-aware Controller Placement for Software-Defined Networks. In *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013) : 27-31 May 2013, Ghent, Belgium*. p. 1418.

Huang, D.Y., Yocum, K., Snoeren, A.C. (2013) High-Fidelity Switch Models for Software-Defined Network Emulation. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM,*.

Huddiniah, E.R., Safitri, E.M., Priyambada, S.A., Nasrullah, M., Angresti, N.D. (2018) Optimasi Rute Untuk Software Defined Networking-Wide Area Network (SDN-WAN) Dengan Openflow Protocol. *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*. **13**(1), 7.

Ibn-Khedher, H., Abd-Elrahman, E., Afifi, H., Forestier, J. (2015) Network issues in virtual machine migration. In *2015 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, pp. 1–6.

Ingale, V., Kakade, S. (2014) *Optimization of Network*. Department of Electrical Engineering, San Jose State university, San Jose, California.

V. Jacobson, R. Braden, D. Borman (1992) TCP extensions for high performance. *RFC 1323*, 1–38. [online]. Available from: https://www.ietf.org/rfc/rfc1323.txt [Accessed May 7, 2017].

Jain, R., Paul, S. (2013) Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*. **51**(11), 24–31.

Jalili, A., Ahmadi, V., Keshtgari, M., Kazemi, M. (2015) Controller placement in software-defined WAN using multi objective genetic algorithm. In *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*. IEEE, pp. 656–662.

Jimenez, Y., Cervello-Pastor, C., Garcia, A.J. (2013) Defining a network management architecture. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*. IEEE, pp. 1–3.

Jimenez, Y., Cervello-Pastor, C., Garcia, A.J. (2014) On the controller placement for designing a distributed SDN control layer. In *2014 IFIP Networking Conference*. IEEE, pp. 1–9.

Jin, H., Pan, D., Liu, J., Pissinou, N. (2013) OpenFlow-Based Flow-Level Bandwidth Provisioning for CICQ Switches. *IEEE Transactions on Computers*. **62**(9), 1799–1812.

Johnson, M. (2009) *Optical Fibres, Cables and Systems*. International Telecommunications Union.

Kaibel, V., Peinhardt, M. (2006) On the bottleneck shortest path problem. *Konrad-Zuse-Zentrum f. Informationstechnik*. (Technical Report), 06-22.

Karypis, G., Eui-Hong Han, Kumar, V. (1999) Chameleon: hierarchical clustering using dynamic modeling. *Computer*. **32**(8), 68–75.

Kaufman, L., Rousseeuw, P.J. (2009) *Finding groups in data : an introduction to cluster analysis*. Vol. 344. John Wiley & Sons.

Kawanishi, T., Kanno, A., Yamamoto, N. (2012) Low latency data transmission using wireless and wired communications. In *International Conference on Space Optical Systems and Applications (ICSOS) 2012*. pp. 1–5.

Kawanishi, T., Kanno, A., Yoshida, Y., Kitayama, K.-I. (2013) Impact of wave propagation delay on latency in optical communication systems. *In Optical Metro Networks and Short-Haul Systems V*. **8646**, 86460C. International Society for Optics and Photo.

Keti, F., Askar, S. (2015) Emulation of Software Defined Networks Using Mininet in Different Simulation Environments. In *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*. IEEE, pp. 205–210.

Killi, B.P.R., Rao, S.V. (2017) Capacitated Next Controller Placement in Software Defined Networks. *IEEE Transactions on Network and Service Management*. **14**(3), 514–527.

Killi, B.P.R., Rao, S.V. (2016) Controller placement with planning for failures in software defined networks. In *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, pp. 1–6.

Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M. (2011) The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications*. **29**(9), 1765–1775.

Koerner, M., Kao, O. (2014) Evaluating SDN based Rack-to-Rack Multi-path Switching for Data-center Networks. *Procedia Computer Science*. **34**, 118–125.

Korkmaz, T., Krunz, M. (2003) Bandwidth-delay constrained path selection under inaccurate state information. *IEEE/ACM Transactions on Networking*. **11**(3), 384–398.

Kreutz, D., Ramos, F.M. V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S. (2015a) Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*. **103**(1), 14–76.

Kuang, H., Qiu, Y., Li, R., Liu, X. (2018) A hierarchical K-means algorithm for controller placement in SDN-Based WAN architecture. In Proceedings - 10th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2018. IEEE, pp. 263–267.

Kuniar, M., Perešíni, P., Kosti, D. (2014) What you need to know about SDN control and data planes. *No. EPFL-REPORT-199497*.

Lange, S., Gebert, S., Spoerhase, J., Rygielski, P., Zinner, T., Kounev, S., Tran-Gia, P. (2015) Specialized Heuristics for the Controller Placement Problem in Large Scale SDN Networks. In *2015 27th International Teletraffic Congress*. IEEE, pp. 210–218.

Lange, S., Gebert, S., Zinner, T., Tran-Gia, P., Hock, D., Jarschel, M., Hoffmann, M. (2015) Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks. *IEEE Transactions on Network and Service Management*. **12**(1), 4–17.

Lara, A. (2015) Using Software-Defined Networking to Improve Campus, Transport and Future Internet Architectures. University of Nebraska-Lincoln, USA.

Leiner, B.M., Cerf, V.G., Clark, D.D., Kahn, R.E., Kleinrock, L., Lynch, D.C., Postel, J., Roberts, L.G., Wolf, S. (2009) A Brief History of the Internet. *ACM SIGCOMM Computer Communication Review*. **39**(5), 22–31.

Li, T., Gu, Z., Lin, X., Li, S., Tan, Q. (2018) Approximation algorithms for controller placement problems in software defined networks. In Proceedings - 2018 IEEE 3rd International Conference on Data Science in Cyberspace, DSC 2018. IEEE, pp. 250–257.

Liao, J., Sun, H., Wang, J., Qi, Q., Li, K., Li, T. (2017) Density cluster based approach for controller placement problem in large-scale software defined networkings. *Computer Networks*. **112**, 24–35.

Liu, J., Liu, J., Xie, R. (2016) Reliability-based controller placement algorithm in software defined networking. *Computer Science and Information Systems*. **13**(2), 547–560.

Loffreda, D. (2015) SDN and NFV: The brains behind the “smart” city | Network World. [online]. Available from: http://www.networkworld.com/article/3000819/software-defined-networking/sdn-and-nfv-the-brains-behind-the-smart-city.html [Accessed January 4, 2016].

Lu, Y.-T. (2003) *An Efficient Hilbert Curve-based Clustering Strategy for Large Spatial Databases*. PhD diss., National Sun Yat-sen University.

Ma, Q., Steenkiste, P. (1998) Routing Traffic with Quality-of-Service Guarantees in Integrated Services Networks. *in Proceedings of NOSSDAV ’98, Cambridge, UK,*.

Márton Zubor , Attila Kőrösi, András Gulyás, G.R. (2014) On the Computational Complexity of Policy Routing. In Y. Kermarrec, ed. Lecture Notes in Computer Science. Cham: Springer International Publishing.

Matlab (2017) Measure node importance - MATLAB centrality - MathWorks United Kingdom. [online]. Available from: https://uk.mathworks.com/help/matlab/ref/graph.centrality.html [Accessed May 6, 2018].

MATLAB (2017) Constructing Sparse Matrices. [online]. Available from: https://uk.mathworks.com/help/matlab/math/constructing-sparse-matrices.html [Accessed October 20, 2017].

Al Mhdawi, A.K., Al-Raweshidy, H.S. (2018) IPRDR: Intelligent power reduction decision routing protocol for big traffic flood in hybrid-SDN architecture. *IEEE Access*. **6**, 10944–10955.

Mijumbi, R., Serrat, J., Gorricho, J., Latre, S., Charalambides, M., Lopez, D. (2016) Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*. **54**(1), 98–105.

Miller, G.L., Teng, S.-H., Vavasis, S.A. (1991) A unified geometric approach to graph separators. In *Proceedings 32nd Annual Symposium of Foundations of Computer Science*. IEEE Comput. Soc. Press, pp. 538–547.

Mininet Team (2016) Mininet Overview. [online]. Available from: http://mininet.org/overview/ [Accessed June 4, 2016].

Muller, L.F., Oliveira, R.R., Luizelli, M.C., Gaspary, L.P., Barcellos, M.P. (2014) Survivor: An enhanced controller placement strategy for improving SDN survivability. In *2014 IEEE Global Communications Conference*. IEEE, pp. 1909–1915.

Nagano, J., Shinomiya, N. (2015) Efficient information sharing among distributed controllers of OpenFlow network with bi-connectivity. In *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, pp. 320–324.

Nakao, A., Yamada, K. (2016) Research and Development on Network Virtualization Technologies in Japan: VNode and FLARE Projects. In *The GENI Book*. Cham: Springer International Publishing, pp. 563–588.

Nakibly, G., Cohen, R., Katzir, L. (2012) *On the Trade-Off between Control Plane Load and Data Plane Efficiency in Software Defined Networks*. Tehnical Report, Technion, Computer Science Department.

Naning, H.S., Munadi, R., Effendy, M.Z. (2016) SDN controller placement design: For large scale production network. In *2016 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*. IEEE, pp. 74–79.

Ng, R.T., Han, J. (1994) Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*. pp. 144–155.

Open Network Foundation (2009) OpenFlow Switch Specification Version 1.0.0. [online]. Available from: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf [Accessed February 18, 2015].

Open Networking Foundation (2013) OpenFlow Switch Specification Version 1.4.0. [online]. Available from: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf [Accessed February 20, 2015].

OpenvSwitch (2016a) ovs-appctl (8) - utility for configuring running Open vSwitch daemons. [online]. Available from: http://openvswitch.org/support/dist-docs/ovs-appctl.8.txt [Accessed July 1, 2016].

OpenvSwitch (2016b) ovs-dpctl (8) - administer Open vSwitch datapaths. [online]. Available from: http://openvswitch.org/support/dist-docs/ovs-dpctl.8.txt [Accessed June 18, 2016].

OpenvSwitch (2016c) ovs-ofctl(8) - administer OpenFlow switches. [online]. Available from: http://openvswitch.org/support/dist-docs/ovs-ofctl.8.txt [Accessed July 2, 2016].

OpenvSwitch (2016d) ovs-vsctl(8) - utility for querying and configuring ovs-vswitchd. [online]. Available from: http://openvswitch.org/support/dist-docs/ovs-vsctl.8.txt [Accessed July 10, 2016].

Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J.J., Lorca, J., Folgueira, J. (2017) Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges. *IEEE Communications Magazine*. **55**(5), 80–87.

Orlin, J.B. (2013) Max flows in O(nm) time, or better. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*. New York, New York, USA: ACM Press, p. 765.

Packard Enterprise, H. (2016a) HPE VAN SDN Controller 2.7 Administrator Guide. [online]. Available from: http://h20564.www2.hpe.com/hpsc/doc/public /display?docId=c05028095 [Accessed July 27, 2016].

Packard Enterprise, H. (2016b) Aruba 3810M Switches Installation and Getting Started Guide. [online]. Available from: https://support.hpe.com/hpsc/doc/public /display?docId=c04948676[Accessed July 27, 2016].

Penna, M.C., Jamhour, E., Miguel, M.L.F. (2014) A Clustered SDN Architecture for Large Scale WSON. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*. IEEE, pp. 374–381.

Perrot, N., Reynaud, T. (2016) Optimal placement of controllers in a resilient SDN architecture. In *2016 12th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, pp. 145–151.

Pfaff, B., Lantz, B., Heller, B. (2012) OpenFlow Switch Specification, version 1.3. 0. *Open Networking Foundation*.

Phemius, K., Bouet, M. (2013) Monitoring latency with OpenFlow. *Network and Service Management (CNSM), 2013 9th International Conference on. IEEE*.

Prieto, I., Izal, M., Magaña, E., Morato, D. (2015) A Simple Passive Method to Estimate RTT in High Bandwidth-Delay Networks. In *The Seventh International Conference on Evolving Internet*.

Pujolle, G. (2015) *Software Networks Virtualization, SDN, 5G and Security*. First. London, UK and Hoboken, USA: ISTE Ltd and John Wiley & Sons, Inc.

Rao, S. (2014) SDN and Its Use-Cases NV and NFV. In *NEC Technologies India Limited*. p. Network, 2, H6.

Rath, H.K., Revoori, V., Nadaf, S.M., Simha, A. (2014) Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, pp. 1–6.

Rosen, K.H. (2012) *Discrete mathematics and its applications*. American Mathematics Competitions (AMC),10(12), p.688-690.

Ruiz-Rivera, A., Chin, K.-W., Soh, S. (2015) GreCo: An Energy Aware Controller Association Algorithm for Software Defined Networks. *IEEE Communications Letters*. **19**(4), 541–544.

Sabidussi, G. (1966) The centrality index of a graph. *Psychometrika*. **31**(4), 581–603.

Sahoo, K.S., Sahoo, S., Sarkar, A., Sahoo, B., Dash, R. (2017) On the placement of controllers for designing a wide area software defined networks. In *TENCON 2017 - 2017 IEEE Region 10 Conference*. IEEE, pp. 3123–3128.

Sallahi, A., St-Hilaire, M. (2017) Expansion Model for the Controller Placement Problem in Software Defined Networks. *IEEE Communications Letters*. **21**(2), 274–277.

Sallahi, A., St-Hilaire, M. (2015) Optimal Model for the Controller Placement Problem in Software Defined Networks. *IEEE Communications Letters*. **19**(1), 30–33.

Sanner, J.-M., Hadjadj-Aoufi, Y., Ouzzif, M., Rubino, G. (2016) Hierarchical clustering for an efficient controllers' placement in software defined networks. In *2016 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, pp. 1–7.

Sans, F., Gamess, E. (2013) Analytical Performance Evaluation of Different Switch Solutions. *Journal of Computer Networks and Communications*. **2013**, 1–11.

Saraee, A., Saraee, M., Ahmadian, N., Narimani, Z. (2007) Data mining process using clustering: a survey Title Data mining process using clustering: a survey Data Mining Process Using Clustering: A Survey. In *Proceedings of IDMC'07*. pp. 1–8.

SDxCentral (2015) SDN and NFV Market Size Report. *SDxCentral*. [online]. Available from: https://www.sdxcentral.com/reports/%0Asdn-nfv-market-size-forecast-report-2015/ [Accessed May 30, 2018].

Service Providers (2017) Network Slicing the Key that Unlocks 5G Revenue Potential - Where 5G meets SDN/NFV. [online]. Available from: https://www.strategyanalytics.com/access-services/service-providers/networks-and-service-platforms/reports/report-detail/network-slicing-the-key-that-unlocks-5g-revenue-potential-where-5g-meets-sdn-nfv#.WnO5Ta5l9aQ.

sFlow.org (2012) sFlow: Software defined networking. [online]. Available from: http://blog.sflow.com/2012/05/software-defined-networking.html [Accessed May 7, 2017].

Sharan, R., Shamir, R. (2000) CLICK: a clustering algorithm with applications to gene expression analysis. *Proceedings International Conference on Intelligent Systems for Molecular Biology*. **8**, 307–16.

Sheikholeslami, G., Chatterjee, S., Zhang, A. (1998) Wavecluster: A multi-Resolution Clustering Approach for Very Large Spatial Databases. *The VLDB Journal : The International Journal on Very Large Data Bases*. **98**, 428–439.

SHESHADRI, S.K. (2004) *Multi-constrained node-disjoint multipath QoS routing algorithms for status dissemination networks*. Doctoral dissertation,Washington State University.

Simon, H.D. (1991) Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*. **2**(2–3), 135–148.

Sinha, D., Haribabu, K., Balasubramaniam, S. (2015) Real-time monitoring of network latency in Software Defined Networks. In *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*. IEEE, pp. 1–3.

Soewito, B., Andy, Gunawan, F.E., Mansuan, M.S. (2017) WAN Optimization to Speed up Data Transfer. In *Procedia Computer Science*. Elsevier, pp. 45–53.

Sonchack, J., Aviv, A.J., Keller, E. (2016) Timing SDN Control Planes to Infer Network Configurations. *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. ACM.*

Sood, K., Xiang, Y. (2017) The controller placement problem or the controller selection problem?. *Journal of Communications and Information Networks*. **2**(3), 1–9.

Stallings, W. (2013) Software-Defined Networks and OpenFlow. *The Internet Protocol Journal,*. **16**(1).

Steenkiste, P., Ma, Q. On path selection for traffic with bandwidth guarantees. In *Proceedings 1997 International Conference on Network Protocols*. IEEE Comput. Soc, pp. 191–202.

Tadinada, V.R. (2014) Software defined networking: Redefining the future of internet in IoT and cloud era. In *Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014*. IEEE, pp. 296–301.

Tanha, M., Sajjadi, D., Pan, J. (2016) Enduring Node Failures through Resilient Controller Placement for Software Defined Networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, pp. 1–7.

Tanha, M., Sajjadi, D., Ruby, R., Pan, J. (2018) Capacity-aware and Delay-guaranteed Resilient Controller Placement for Software-Defined WANs. I*EEE Transactions on Network and Service Management*, 1–1.

Tim De Backer (2014) Use Linux Traffic Control as impairment node in a test environment. [online]. Available from: https://www.excentis.com/blog/use-linux-traffic-control-impairment-node-test-environment-part-1.

Tootoonchian, A., Ghobadi, M., Ganjali, Y. (2010) OpenTM: Traffic Matrix Estimator for OpenFlow Networks. In A. Krishnamurthy & B. Plattner, eds. *11th Int. Conf. Passive Active Meas*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, p. 201–210.

Trimintzios, P., G. Pavlou, Andrikopoulos, I. (2000) Providing Traffic Engineering Capabilities in IP Networks Using Logical Paths. *, Eighth IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks (IFPM ATM and IP 2000), Ilkey, UK July (2000)*.

Tseng, L.Y., Yang, S.B. (2001) A genetic approach to the automatic clustering problem. *Pattern Recognition*. **34**(2), 415–424.

Tuncer, D., Charalambides, M., Clayman, S., Pavlou, G. (2015a) Adaptive Resource Management and Control in Software Defined Networks. *IEEE Transactions on Network and Service Management*. **12**(1), 18–33.

Tuncer, D., Charalambides, M., Clayman, S., Pavlou, G. (2015b) On the placement of management and control functionality in software defined networks. In *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, pp. 360–365.

Turull, D., Hidell, M., Sjodin, P. (2014) Performance evaluation of openflow controllers for network virtualization. In *2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, pp. 50–56.

Van, J., Craig, L., Steven, M., Lawrence Berkeley National Laboratory, University of California, Berkeley, CA (2016) Manpage of TCPDUMP. [online]. Available from: http://www.tcpdump.org/tcpdump_man.html [Accessed July 2, 2016].

Veness, C. (2016) Vincenty solutions of geodesics on the ellipsoid. *Movable Type Scripts*. [online]. Available from: https://www.movable-type.co.uk/scripts/latlong-vincenty.html [Accessed May 15, 2017].

Walshaw, C.H., Cross, M., Everett, M.G. (1995) A Localized Algorithm for Optimizing Unstructured Mesh Partitions. *The International Journal of Supercomputer Applications and High Performance Computing*. **9**(4), 280–295.

Wang, G., Zhao, Y., Huang, J., Duan, Q., Li, J. (2016) A K-means-based network partition algorithm for controller placement in software defined network. In *2016 IEEE International Conference on Communications (ICC)*. IEEE, pp. 1–6.

Wang, G., Zhao, Y., Huang, J., Wu, Y. (2018) An Effective Approach to Controller Placement in Software Defined Wide Area Networks. *IEEE Transactions on Network and Service Management*. 15(1), 344–355.

Wang, J., Zhou, M., Li, Y. (2004) Survey on the End-to-End Internet Delay Measurements. In Springer, Berlin, Heidelberg, pp. 155–166.

Wang, S.-Y. (2014) Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet. In *2014 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, pp. 1–6.

Wang, S., Zhang, J., Huang, T., Liu, J., Liu, Y., Yu, F.R. (2017) FlowTrace: measuring round-trip time and tracing path in software-defined networking with low communication overhead. *Frontiers of Information Technology & Electronic Engineering*. **18**(2), 206–219.

Wang, T., Liu, F., Guo, J., Xu, H. (2016) Dynamic SDN controller assignment in data center networks: Stable matching with transfers. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. IEEE, pp. 1–9.

Wang, T., Liu, F., Xu, H. (2017) An Efficient Online Algorithm for Dynamic SDN Controller Assignment in Data Center Networks. *IEEE/ACM Transactions on Networking*. **25**(5), 2788–2801.

Wang, W., Yang, J., Muntz, R. (1997) STING : A Statistical Information Grid Approach to Spatial Data Mining. In *The 23rd VLDB Conference Athens*. pp. 186–195.

Williams, R.D. (1991) Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience*. **3**(5), 457–481.

Wireshark (2016a) tshark - The Wireshark Network Analyzer 2.0.0. [online]. Available from: https://www.wireshark.org/docs/man-pages/tshark.html [Accessed July 12, 2016].

Wireshark (2016b) Wireshark Go Deep. [online]. Available from: https://wireshark.org/ [Accessed July 3, 2016].

Wu, S. (2012) *Data Fusion in Information Retrieval*. Springer Publishing Company, ed. Springer Publishing Company.

Xiao, P., Li, Z., Guo, S., Qi, H., Qu, W., Yu, H. (2016) A K self-adaptive SDN controller placement for wide area networks. *Frontiers of Information Technology & Electronic Engineering*. **17**(7), 620–633.

Xiao, P., Qu, W., Qi, H., Li, Z., Xu, Y. (2014) The SDN controller placement problem for WAN. In *2014 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, pp. 220–224.

Xu, R., WunschII, D. (2005) Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*. **16**(3), 645–678.

Yalda, K.G., Hamad, D.J., Okumus, I.T. (2015) Design and Implementation of an Intra-domain routing module for an SDN controller for Traffic Engineering in SDN environment. In *2015 International Conference on Advances in Software, Control and Mechanical Engineering (ICSCME-2015) Sept. 7-8, 2015 Antalya (Turkey)*. p. 93.

Yanyu Chen, Qing Li, Yuan Yang, Qi Li, Yong Jiang, Xi Xiao (2015) Towards adaptive elastic distributed Software Defined Networking. In *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, pp. 1–8.

Yao, G., Bi, J., Li, Y., Guo, L. (2014) On the Capacitated Controller Placement Problem in Software Defined Networks. *IEEE Communications Letters*. **18**(8), 1339–1342.

Yao, L., Hong, P., Zhang, W., Li, J., Ni, D. (2015) Controller placement and flow based dynamic management problem towards SDN. In *2015 IEEE International Conference on Communication Workshop (ICCW)*. IEEE, pp. 363–368.

Yu, C., Lumezanu, C., Sharma, A., Xu, Q., Jiang, G., Madhyastha, H. V (2015) Software-defined Latency Monitoring in Data Center Networks. *International Conference on Passive and Active Network Measurement. Springer International Publishing*, 360–372.

Zeng, D., Teng, C., Gu, L., Yao, H., Liang, Q. (2015) Flow setup time aware minimum cost switch-controller association in software-defined networks. In *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), 2015 11th International Conference*. p. 259–264. IEEE.

Zhang, A. (2006) *Advanced analysis of gene expression microarray data*. 1st ed. World Scientific Publishing Company.

Zhang, B., Wang, X., Huang, M. (2018) Multi-objective optimization controller placement problem in internet-oriented software defined network. *Computer Communications*. 123, 24–35.

Zhang, T., Bianco, A., De Domenico, S., Giaccone, P. (2016) The Role of Inter-Controller Traffic in the Placement of Distributed SDN Controllers. *CoRR,vol. abs/1605.09268*.

Zhang, T., Bianco, A., Giaccone, P. (2016) The role of inter-controller traffic in SDN controllers placement. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, pp. 87–92.

Zhang, T., Ramakrishnan, R., Livny, M., Zhang, T., Ramakrishnan, R., Livny, M. (1996) BIRCH. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data - SIGMOD '96*. New York, New York, USA: ACM Press, pp. 103–114.

Zhang, Y., Beheshti, N., Tatipamula, M. (2011) On Resilience of Split-Architecture Networks. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*. IEEE, pp. 1–6.

Zhu, L., Chai, R., Chen, Q. (2017) Control plane delay minimization based SDN controller placement scheme. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, pp. 1–6.

Zilong Ye, Patel, A.N., Ji, P.N., Chunming Qiao, Ting Wang (2013) Virtual infrastructure embedding over software-defined flex-grid optical networks. In *2013 IEEE Globecom Workshops (GC Wkshps)*. IEEE, pp. 1204–1209.

Zoher Bholebawa, I., Dalal, U.D. (2016) Design and Performance Analysis of OpenFlow-Enabled Network Topologies Using Mininet. *International Journal of Computer and Communication Engineering 5.6*, 419.

Zumbusch, G. (2000) *On the Quality of Space-Filling Curve Induced Partitions*. Sonderforschungsbereich 256.