Scalable Reliable Controller Placement in Software Defined Networking

Abdunasser Alowa

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented In Partial Fulfillment of The Requirements

For The Degree of

Doctor of Philosophy (Computer Science) at

Concordia University

Montréal, Québec, Canada

September, 2020

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Abdunasser Alowa**

Entitled: **Scalable Reliable Controller Placement in Software Defined Networking**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science) at

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair
Dr. Rajamohan Ganesan

_____ External Examiner
Dr. Elhadi Shakshuki

_____ Examiner
Dr. Anjali Agarwal

_____ Examiner
Dr. Hovhannes Harutyunyan

_____ Examiner
Dr. Dhrubajyoti Goswami

_____ Supervisor
Dr. Thomas Fevens

Approved by _____
Dr. Nicola Pezolet, Graduate Program Director

01 Sep 2020 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Scalable Reliable Controller Placement in Software Defined Networking

Abdunasser Alowa, Ph.D.

Concordia University, 2020

Software Defined Networking (SDN) is a new networking paradigm that facilitates a centralized system of computer networks by decoupling the control and data plane from each other, where a controller maintains the management of a global view of the network. SDN architectures can provide programmatic interfaces in communication networks that significantly simplify network management. Hence, the controllability and manageability of a network can be improved. On the one hand, the placement of controllers can significantly impact network performance in terms of controller responsiveness. On the other hand, SDN offers the ability to have controllers distributed over the network to solve the single point of failure problem at the control plane, increasing scalability and flexibility. However, there are some inevitable problems for such networks, especially for controller-related problems. For instance, scalability, reliability, and controller availability are some of the hottest aspects of SDN. More precisely, failure of the controllers themselves may lead to the impact of these aspects and the collapse of the network performance.

Despite the issues mentioned above, the controller placement challenges must be appropriately addressed to take advantage of the SDN. The connections between the controller (control plane) and the switches (data plane) in SDN are established by either an in-band or an out-of-band control mechanism. New challenges still arise regarding the connection availability and provide more protection for the connection between the data and control planes. A disconnection between the two planes could result in performance degradation. Although the SDN offers the advantage of an environment of multiple distributed controllers, yet the intercommunication factor between these controllers is still a key challenge. This thesis investigates the issues mentioned above and organizes them into four stages.

First, dealing with the controller placement problem as the most crucial concern in SDN, via exploiting the independent dominating set approach to ensure a distribution of controllers with lowest response times. We propose a new node degree-based algorithm named High Degree with Independent Dominating Set (HDIDS) for the controller placement problem in the SDN networks. HDIDS is composed of two phases to deal with controller placement: (1) determining candidate

controller instances by selecting those nodes with the highest degree; and (2) partitioning the network into multiple domains, one controller per domain.

To further improve network performance, reliability, and survivability, one solution is to deploy backup controllers to satisfy the quality of service requirements. In this regard, as a second step, we enhance the controller placement approach by designing a reliable and survivable controller placement strategy. This strategy relies on the efficient deployment of backup controllers by constructing virtual backup domains set(s) to ensure the durability and resilience of network control management. The approach design is called a Survivable Backup Controller Placement approach.

Furthermore, to achieve reliable control traffic between data and control planes in an in-band control network, as a third stage, we design and implement an In-band Control Protection Module that finds a set of ideal paths for the control channel under the failure conditions. The proposed protection mechanism protects as much control traffic as possible.

Finally, we present a practical approach for the controller placement problem in software defined networks aiming to minimize the inter-controller communication delay time and the delay time between controller and switches. The principal concept employed in this approach is the Connected Dominating Set. Further, we present an algorithm using the Minimum Connected Dominating Set, which minimizes the delay time between the distributed SDN controllers.

Acknowledgments

First, I would like to thank Almighty Allah for giving me the opportunity, determination, and strength to do my research. His continuous grace and mercy was with me throughout my life and ever more during the tenure of my research.

I would like to express my gratitude to my supervisor, Dr. Thomas Fevens whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and skill in many areas and his assistance in writing reports. I doubt that I will ever be able to convey my appreciation fully, but I owe him my eternal gratitude.

I would like to thank the members of my committee for their advice they provided at all levels of the research project.

Also, I would like to extend my deepest appreciation to all the staff in the Department of Computer Science and Software Engineering at Gina Cody School of Engineering and Computer Science for kind support and help on the technical and administrative aspects of the study.

A special thanks for the Ministry of Higher Education and Scientific Research, Libya for financially supporting my research, and granting me the opportunity to study abroad.

Nobody has been more important to me in the pursuit of this project than the members of my family. A very special thanks go out to my loving and supportive wife Fatima, and my wonderful children (Mabrouka, Mayar, Lujain, and Ahmed), without their love, encouragement, and patience, I would not have finished this thesis.

Most importantly, I would like to thank my parents, whose love and guidance are with me in whatever I pursue in my general life. They are the ultimate role models.

I never forget my special thanks to my wife's mother for praying each moment for me to be successful.

Finally, my grateful thanks to my friends who assist me even with a piece of advice.

Really, thank you all „„

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
ART	Average Response Time
BFD	Bidirectional Forwarding Detection
BFM	Backup Forwarding Mode
BGS	Backup Gate Switch
CCSM	Control Channel Status Monitoring
CDS	Connected Dominating Sets
CNP	Control Network Protection
CPDR	Control Packet Drop Rate
CPP	Controller Placement Problem
CPRR	Control Packet Receive Rate
CR	Control Recovery
DS	Dominating Set
DS	Detector Switch
FE	Full Enumeration
FRT	Failure Recovery Time
FS	Failed Switch
HBM	Hop-Bypass Mode
HD	High Degree
HDIDS	High Degree with Independent Dominating Set
ICPM	In-band Control Protection Module
IDASD	Inter-Domain Adjacent with Short Distance
IDS	Independent Dominating Sets
ILP	Integer Linear Programming
IT	Information Technology
LDSD	Low Degree with Short Distance
LLDP	Link Layer Data Packets
MCDS	Minimum Connected Dominating Set
MDPC	Modified Density Peaks Clustering
MDSD	Max Degree with Short Distance
MGS	Master Gate Switch
MPLS	Multi Protocol Label Switching
MRT	Maximum Response Time
NFV	Network Functions Virtualization

NOS	Network Operating System
NSS	Neighboring Switches Statistics
OF	OpenFlow
PS	Protector Switch
QoS	Quality of Service
SDN	Software Defined Networking
SPoF	Single Point of Failure (SoPF
TLS	Transport Layer Security
VBD	Virtual Backup Domain
WAN	Wide Area Networks
WCDS	Weakly Connected Dominating Sets

Chapter 1

Introduction

1.1 Software Defined Networking in glance

Network technologies have helped to solve many computing challenges. They have enabled the successful development of a variety of emerging technologies, such as distributed cloud computing. However, the traditional approaches to innovation in network technologies can be very slow, complicated, inflexible, and expensive. This has contributed to the slow development of scalable information technology (IT) infrastructure and a diversity of IT solution issues.

There is a need for new solutions that address the problems with traditional network technologies. These solutions will facilitate the development of flexible and cost-effective network services and applications. Recently, Software Defined Networking (SDN) and Network Functions Virtualization (NFV) have emerged as two complementary technologies for enabling flexible and programmable networking. SDN makes the control of networking programmable; this allows elastic networks that can respond to changing needs. Until recently, typically network layers were merged on the same device.

To meet the varied needs of the industry, service providers, organizations, and even end-users, various functional components of today's network resources are separated physically and operationally from each other. Although the traditional network architecture has worked fine, with current hardware Virtualization common today, it becomes challenging, if not impossible, for the conventional network architecture to meet the new virtual requirements. A network device is comprised of a data plane that is often a set of switches connecting the different network ports on a device, and a control plane that is the brain of the device. For instance, to set up network paths, all routing protocols used to build loop-free paths inside a network are executed in a distributed fashion. That is, each device in the network has a separate control layer that implements the routing protocol. However,

in a centralized control plane model, only one (or at least logical) control plane exists. This brain pushes commands to every device and thus drives them to manipulate its physical switching and routing hardware, typically in networks' layers were merged on the same device.

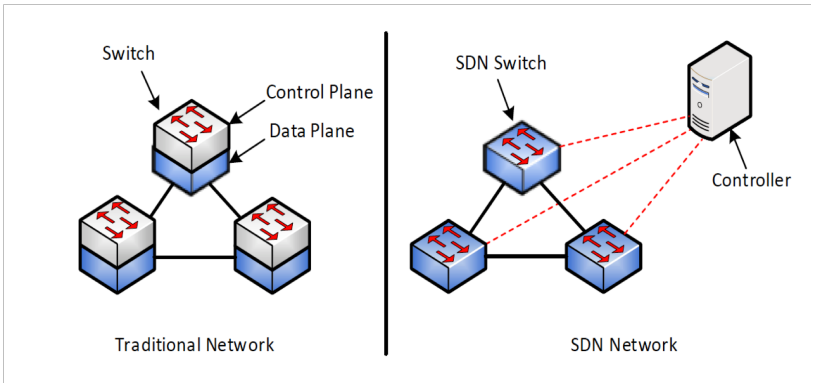

Figure 1: Traditional network view Vs SDN Network view

As shown in Figure 1, in traditional networks, the control and data planes are combined in a network device. The control plane is responsible for the configuration of the node and the programming of the paths to be used for data flows. After the paths have been determined, as in forwarding policy, they are pushed down to the data plane. Policy adjustment is possible only through changes to the configuration of the devices. This design might impose restrictions for network operators who wish to scale their networks in response to changing traffic demands.

Differently, on the right side of Figure 1, in the SDN approach, the control plane is moved out of the individual network devices and into a separate, centralized controller. SDN switches are managed by a network operating system (NOS). The NOS collects information using an Application Programming Interface (API) and manipulates the data plane of the switches. It also provides an abstract model of the network topology to the SDN controller. With this design, the controller can exploit the complete knowledge of the network to improve flow management and support service-user requirements of scalability and flexibility [2]. Although network operators can build and utilize network control mechanisms with different SDN protocols, some issues regarding scalability, reliability, and robustness remain. In the multiple distributed SDN networks, it is not easy to change the switches and protocols configurations for network managers. Therefore, the need for effective mechanisms to make the network stable and robust is inevitable. At the same time, the design of these mechanisms must be done without sacrificing the flexibility of network control. In addition, to satisfy these criteria, as the controller plays a key role in SDN networks, several controller-related factors can significantly impact on the network performance. For instance, the placement of the controller is important to keep a network controllable and manageable. This thesis discusses mechanisms for address and avoidance of stability and robustness issues in SDN networks.

1.2 Research Focus

Despite the efficient solutions for utilizing SDN, there are some open issues and challenges that should be addressed. Multiple distributed controllers in a large-scale SDN environment is our main objective in this research. More specifically, we are interested in different aspects of SDN as follow:

1.2.1 Controller Placement Problem in SDN

The controllers are the brain of SDN and the functions major engine; hence, controller placement can greatly impact network performance in terms of controller responsiveness. In order to address the controller placement problem and allow for distributed and scalable solutions, controllers need to communicate with their switches such that routing updates, controller allocation information, etc., can be shared with them promptly. The response time between the distributed controllers is also an important factor. Dealing with this problem can be simplified by considering two questions: how many controllers are needed to manage the whole network, and where should they be placed in the network? [3]. The best placement of controllers can be achieved in terms of different metrics and considerations based on the objectives of the network. Moreover, the response time of switch-controller and inter-controller must be satisfied. Such delay time thresholds are of utmost importance since they impact the control decision. For this reason, getting the best possible placement requires studying new and efficient methods and metrics to determine the impact of the placement of controllers on SDN performance.

1.2.2 Survivability in SDN

The core concept of SDN is to abstract the control plane from the data plane. SDN architectures can provide programmatic interfaces in communication networks that significantly simplify network management and improves the efficiency of utilization. Distributed multiple controller SDN environments have become the preferred solution towards better scalability of SDNs. As known, a chance of failure in the network exists. For instance, each controller should handle a certain number of requests of their assigned switches due to its resource constraints. Therefore, the controller will be subject to failure if traffic demands exceed the controller's capacity. Moreover, a controller may fail due to software or hardware issues. Thus, survivability of the control network in case of controller failure recovery is a crucial concept to maintain the continuity of the network operations and to meet the Quality of Service (QoS) requirement in SDN networks. To this end, in the thesis, we intend to study the possibility of controller survivability in order to maintain reliable network performance.

1.2.3 In-band Control Failure in SDN

The control channel in SDN constructed in one of two ways; In-band or Out-of-band. Most of the earlier research works have extensively studied the out-of-band scheme. Therefore, in our research, we focus on the in-band approach, where all control messages are exchanged through the same data-plane channels, unlike the out-of-band, which uses separate channels for each switch. In in-band SDN networks, this matter is crucial, especially when there are unexpected issues raised. For instance, the in-band control scheme enforces the existence of the dependency networks pattern, where a set of nodes in the network rely on sending the control traffic over a particular node. Therefore, dealing with control channel recovery in failure event is one of the in-band network challenges that need to be investigated. Motivated by these pointers, we aim to cope with in-band failure occurrence in SDN effectively. Mainly, providing more protection and controllability of the in-band SDN network under the failure scenario is one of our objectives in the research.

1.2.4 Inter-controller Delay Time Minimization in SDN

With respect to scalability and using multiple SDN controllers, that assist in employing a larger number of network switches, intercommunication cost between distributed controllers is still a key challenge. Under this part of the thesis, an effective approach for the controller placement problem in SDN that aims to minimize the communication delay time between the controllers and the delay time between controller and switch is addressed.

1.3 Research Contributions

With SDN evolving, its promise is clear. Significantly, most of communication service providers accelerate the applications deliver time and enhancing the quality of these services. Despite its advantages, SDN brings up more challenges that need more investigation and analysis. One of the major concerns about SDN is the delay time among all network components, whether between the elements of control and data planes or between the elements in the same plane. Moreover, satisfying the reliability, scalability, and performance requirements, considering the various aspects of SDN, requires more research. Motivated by these concerns, this research focus on improving SDN taking into account different criteria. We summarized and concluded the key contributions of this research as follows: One of the most various factors which affect the scalability of SDN is the placement of controllers. The best placement of controllers can be achieved in terms of different metrics and considerations based on the network's objectives. For this reason, getting the best possible placement requires studying new and efficient methods and metrics to determine the impact of the placement

of controllers on SDN performance.

Therefore, in **Chapter 3**, we present a dynamic approach to ensure the distribution of the controllers and defines the number of controllers required to satisfy a required maximum response time in the network. This algorithm implemented based on the Independent Dominating Set concept combined with the node's degree, in which the node with high connected links will be selected as the base of defining the controller's placement in the network. Compared to the K-means algorithm [4], we conduct extensive simulation experiments to show and evaluate our approach performance. The related paper:

- **Abdunasser Alowa**, Thomas Fevens, Combined degree-based with independent dominating set approach for controller placement problem in software defined networks. In *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN 2019, Paris, France)* - ***Published.***

In the distributed multiple controller SDN architectures, backup controllers, will be instantly assigned to the switches previously managed by the failed controller. However, one controller's failure can greatly affect the backup controller due to the added load (overload), and in the worst-case scenario, generating consecutive failures to other controllers leading to the crash of the entire network. Therefore, and to construct an effective and reliable SDN control network, in **Chapter 4**, we propose a survivable backup controller placement approach to provide more efficient protection for the primary control plane. Our mechanism inspired by building a virtual backup domain aiming to finds the best locations of the backup controllers for fast recovery from primary controllers' failures. The related paper:

- **Abdunasser Alowa**, Thomas Fevens, Yaser Khamayseh, Survival Backup Strategy for Controller Placement Problem in SDN. In *International Journal of Computer and Telecommunications Networking (Computer Networks - 2020)* -***Under Review.***

Despite all the advantages of SDNs, new challenges arise regarding the connection availability between the data and control planes. A disconnection between the two planes could result in performance degradation, especially if the control traffic is affected. Motivated by these concerns, in **Chapter 5**, we propose and implement an in-band control protection approach to improve the robustness and stability of the control channel between the control and data planes in SDN. Our module has demonstrated clear superiority in the performance of finding the ideal paths for in-band control channel recovery, whereas much control traffic as possible can be protected. The related paper:

- **Abdunasser Alowa**, Thomas Fevens, A Dynamic Recovery Module For In-band Control Channel Failure In Software Defined Networking. In *IEEE Conference on Network Softwarization and Workshops (NetSoft 2020)- Ghent, Belgium* - ***published.***

When SDN is deployed in large-scale networks, they may consist of multiple domains, each with sets of switches managed by a single controller. Due to this distributed design layout, The controller placement should satisfy performance requirements such as the maximum allowable delay time of inter-controller communication for synchronization purposes as well as the delay time between the controllers and its assigned switches. We formulate the controller placement problem taking into consideration the delay times of inter-controller and controller-switch. To solve this problem, in **Chapter 6**, we employ the Minimum Connected Dominating Set (MCDS) strategy to improve the distribution of the controllers in the network, achieving a low average delay time between the controllers while maintaining an acceptable average delay time between the controllers and their assigned switches. The evaluation shows that our proposed mechanism effectively decreases the inter-controller delay time as well as maintaining the controller-switch delay time at an acceptable level. The related paper:

- **Abdunasser Alowa**, Thomas Fevens, Towards Minimum Inter-Controller Delay Time in Software Defined Networking. In *15th International Conference on Future Networks and Communications (FNC 2020)- Leuven,Belgium* - ***Published.***

1.4 Research Outline

The thesis is organized as follows:

- **Chapter 1** presents a glance about the networks and the early solutions that laid the foundation for SDN, followed by research objectives, research contribution, and end up by an overview of the thesis structure.

- **Chapter 2** explores the SDN architecture and its benefits, challenges. Moreover, this chapter provides an overview of OpenFlow protocol characteristics. Then we show the structure of the SDN control plan where we describe using a single controller versus multiple controllers; in addition, we outline in-band and out-f-band control architecture in SDN.

- **Chapter 3** puts focus on addressing the controller placement problem CPP in SDN. In this chapter, we describe High Degree with Independent Dominating Set (HDIDS) approach. We explain our method philosophy in detail and compare it with other previous works in the same domain, followed by all experiments results.

- **Chapter 4** explains how to improve the survivability of the controller in SDN. In this chapter, we address the failure occurrence problem in SDN, model this NP-hard problem based on virtual backup domain VBD, and solve it using our proposed heuristics.

- **Chapter 5** discusses the in-band control channel failure in SDN, how to achieve reliable control traffic between data and the control planes. Therefore, we present an In-band Control Protection Module (ICPM), which provides a protection approach of the in-band control channel in SDN.

- **Chapter 6** researches a different direction of controller placement problem. In this chapter, we discuss the controller placement problem in terms of minimizing the inter-controller delay time and its influence on determining the number of required controllers and their locations in the network.

- **Chapter 7** summarizes and concludes my thesis, followed by the potential directions of the research in the future.

Chapter 2

Background

2.1 Software Defined Networking

This chapter provides an overview of SDN and OpenFlow. Furthermore, we explorer different directions of the SDN controller architecture. Finally, we shed light on the control channel scheme in SDN networks.

2.1.1 SDN Architecture

The main idea behind SDN is to make the next generation of networks more dynamic, adaptive, and manageable to changing network requirements easily and rapidly. SDN is suitable for a network where each host is connected and managed by a single administrator e.g., a campus, a company, or a data center network. It is hard to make traditional networks dynamic and adaptive because network control software is implemented in the network devices. The network administrators must ask the network device vendors to implement new control mechanisms to meet their new needs any time a new network service or configuration automation mechanism is introduced into their networks. SDN was developed by the Open Networking Foundation (ONF). The organization is funded by technology companies, including Microsoft, Google, Facebook, Verizon, and Yahoo!. ONF seeks to improve networking through SDN, creates SDN standards such as the OpenFlow standard and related technologies. SDN introduces a separation between the data plane and the control plane. This separation makes it possible to remove control decisions from network hardware, allows for programmable network hardware, and for the creation of software that defines the behavior of the network [5].

Figure 2 shows the SDN architecture that is composed of three planes defined as follows:

1. **Application Plane**

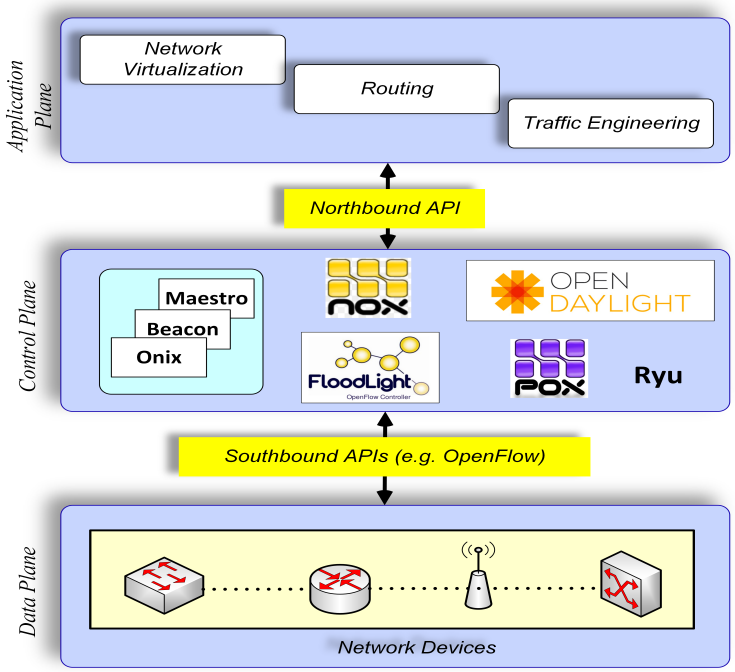

Figure 2: SDN Architecture

This plane hosts a variety of applications for the network (e.g., routing, virtualization, traffic management). The applications can implement an abstracted view of the network by communicating with the controller using an open application programming interface (API).

2. **Northbound API**

   This channel is the API that the application plane uses to communicate with the controller plane. Its primary function is to enable communication between a particular component of a network in the control layer and a higher-level component in the application layer.The applications plane can send to the network controller what is their needs (e.g. data, storage, bandwidth, and so on) and the controller can deliver those resources.

3. **Control Plane**

   This plane is the brain of the SDN architecture and represents a status view of the complete network. The controller is responsible for translating all devices requirements that are directed from the application plane down to the data plane by populating the routing information in the forwarding tables of the data plane elements. Moreover, the controller is in charge of making decisions on how to forward the packets between network devices and pushing such decisions to the network devices for execution [6].

4. **Southbound API**

   The Southbound API allows the communication between the SDN control plane and the data plane elements such as network nodes (both physical and virtual switches) by using a communication protocol. The most well-known protocol used by this API is OpenFlow [7].

5. **Data plane**

   This plane is comprised of switches and routers that perform the function of forwarding data packets from one device to another based on the information in the forwarding tables that are programmed by the control plane protocols.

2.1.2 SDN Benefits

SDN seeks to separate the control plane from the data plane. While the control plane decides how to deal with the traffic, the data plane forwards traffic based on the decisions that the control plane makes. This separation of the control plane by SDN leads to the control of multiple data plane elements (i.e., routers, switches, and other devices) via a single software control program. Other advantages of SDN are as follows:

1. Centralized network provisioning

   SDN architecture grants the capability of control network management from a centralized perspective. In a nutshell, by separating data and control planes, SDN allows the user to provision physical and virtual elements from one location. Although, in traditional networks, managing numerous disparate systems is difficult due to the individual infrastructure monitoring task. SDN eliminates this barrier and allows an administrator to drill among network layers and adapts with the network purposes [8].

2. Lower operating costs

   SDN is one of the most promising new generation networking technologies. By adopting this layout, network operators have more potential to control their infrastructure, allowing customization, optimization, and improving network performance, efficiently. Consequently, reducing overall capital and operational costs.

3. Hardware savings and reduced capital expenditures

   Adopting SDN also gives the administrator the ability to optimize the network hardware efficiently. The control component instructions can improve re-using existing equipment with a new purpose at will. Hence, less expensive hardware can be deployed to archive a high effect.

4. Cloud abstraction

   Cloud computing has become widely accepted and continues to evolve into a unified computing

infrastructure. Using SDN to abstract cloud resources, it is easier to unify the resources. The SDN controller can manage all the various network components that make up the data center platforms.

5. Guaranteed content delivery

   One of the main benefits of SDN is the ability to manage and control data traffic. This can facilitate the implementation of quality of services (QoS) for voice over IP (VOIP) and multimedia transmissions. The advantages of SDN vary from network to network. Thus, it is important to assess the network components and infrastructure to determine whether SDN can help to address issues related to resource availability, virtualization, and network security.

2.1.3 SDN Challenges

In this section, I discuss the weaknesses and challenges of dealing with the SDN networks and the OpenFlow protocol. SDN and OpenFlow offer a way to simplify the prototyping, deployment, and management of the network elements. However, some aspects can affect the safety and the availability of networks and must be taken into consideration. These aspects are discussed below.

1. Availability

   One of the essential aspects that must be considered. The strong dependence between the switches and the controller can be a problem whenever a modification of the rules is needed. Furthermore, if there is only one centralized controller in the network, the controller might become a single point of failure (SPoF). To guarantee controllers' availability and failure recovery, a distributed approach can be implemented.

2. Scalability

   A common perception that the centralized controller in SDN and with increasing the number of switches, flows, bandwidth, etc. may fail to handle all the incoming requests while providing the same quality of service. Therefore, a variety of factors (e.g., the controller's capacity, the placement of the controllers, load balance, and the latency between controllers and network devices) that would help in the process of scale-up the networks while maintaining the standard of providing high-quality service within an acceptable time level. Moreover, these considerations can facilitate a stable long-term networking environment.

3. Resilience

   In traditional networks, when the failure happens to one of the network's switch, the backup paths which are pre-programmed into neighboring switches will be activated. Thus, traffic can switch to one of the backup paths. The core feature of the SDN approach is the decoupling

between the data plane and the control plane. Therefore, if a failure occurs, it would affect both control and data planes. In other words, when a switch fails in the network, it not only affects the switch-to-controller communication but also all the switches along the control path, including the failed switch. On the other hand, if the control logic in SDNs networks fails, the ability of management controlling and forwarding will break down. This will result in undelivered data packets, the dropping of flow requests, and an unreliable network. Beheshti *et al.* in [9], for example, considered the connection resiliency between the controller and the switches. Their proposed algorithms were designed to increase the possibility of fast failover based on resilience-aware controller placement and routing of the control's traffic.

4. Security

   In SDN, the fact that the controller has the essential knowledge of the network makes it a potential target of attacks and threats, especially, if the control component is a single. Also, the channels among the controller and the switches could be vulnerable. According to the OpenFlow specification, it is possible to use secure communication with the TLS protocol, but this depends on the design of the network.

5. Flow table consistency

   The controller can directly modify flow tables on different OpenFlow switches. Thus, a network update leads to an update of the flow tables. In other words, the packets matching must be in the same way in all OpenFlow flow tables. Otherwise, a difference in the flow tables entries may lead to inconsistency of OpenFlow tables, which will lead to false network decisions. To avoid such network update issues, the controller must guarantee efficient flow tables consistency.

6. The Network performance

   The control model adopted can influence network performance. Because the flow table size is limited, the management of a vast number of flows can present some performance challenges. However, a well-designed network could reduce performance issues through a proactive approach, which is known to achieve better performance than the reactive mode because it limits the number of messages exchanged between the controllers and the switches.

2.2 OpenFlow

This section will shed light on the most well-known protocol utilized to initiate the connection between the control and the data plans, that is, the OpenFlow protocol.

2.2.1 OpenFlow Overview

OpenFlow is the most widely used communication protocol for the interaction between the control and data layers in SDN [7]. OpenFlow allows direct access and easy manipulation of the data plane elements such as switches and routers, both physical and virtual. It was designed for network traffic management between two different planes. A protocol like OpenFlow is needed to move network control out of the networking switches to logically centralized control software.

OpenFlow-enabled switch

An OpenFlow-enabled switch consists of one or more flow tables and a group table, various actions such as *add, delete, and update* of flow entries can be done by the controller. OpenFlow switches exchange the control messages with the controller through a specific channel. Figure 3 illustrated the OpenFlow-enabled switch layout. When the OpenFlow switch receives a packet, and it has no

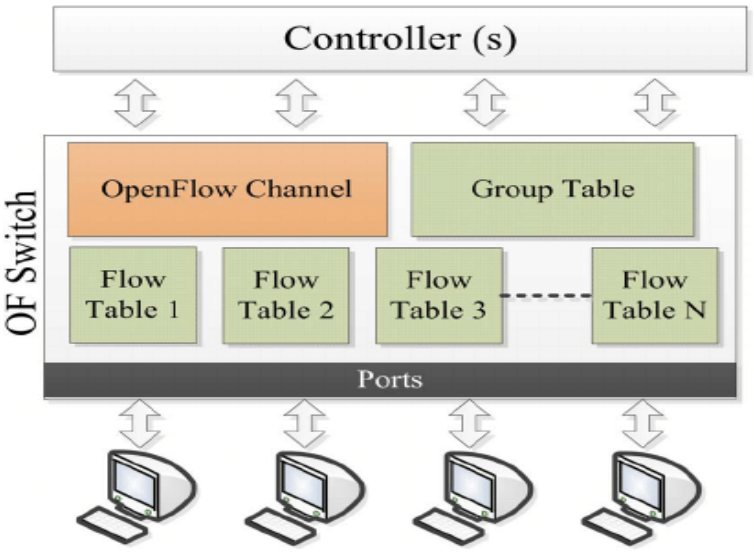

Figure 3: OpenFlow-enabled Switch

match in the flows entries, the switch takes action by sending this packet to the controller. In its turn, the controller then makes a decision on how to handle the packet by either dropping or adding it as a new flow entry and instructing the switch on how to forward similar packets in the future [10].

Packet matching

When a packet arrives at the Flow Table, the packet match fields that can be different according to the packet type are extracted from the packet header, and they are used for the table lookup.

Furthermore, the matching process may differ depending on the data received from the ingress port and, in case the metadata fields, that can also be used to pass information between tables. Thus, in case of positive matching, an instruction set associated with the matched flow entry will be executed by the switch. These instructions typically can contain actions (like packet forwarding, packet modification, and group table), or they can modify the pipeline processing. Furthermore, if some measures are applied during the pipeline processing, the changes are reflected in the packet match fields, which represent the current packet state. In any case, when the instruction set associated with a matching flow entry does not specify the next table, the pipeline processing stops. Only at that time, the packet is processed with its associated actions set and usually forwarded, as shown in Figure 4. More details on all other mechanisms of OpenFlow-enabled switches, such as pipeline processing, pipeline consistency, tables instructions, etc., can be found in [10].

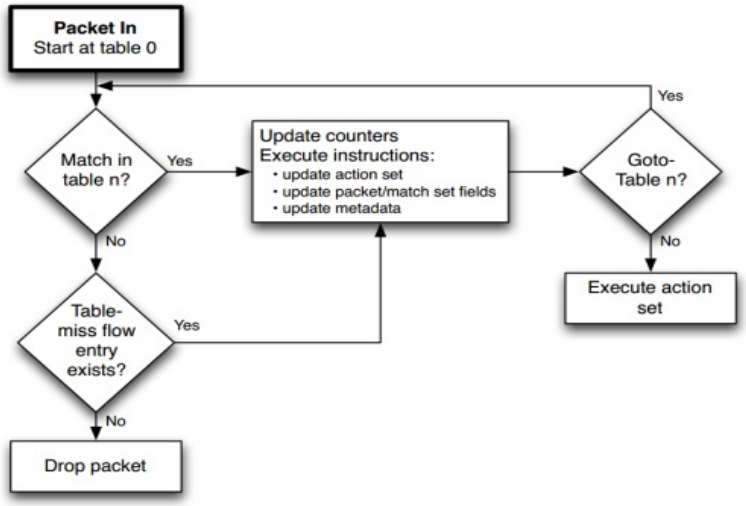

Figure 4: OpenFlow Packets Matching Flowchart

2.2.2 OpenFlow Protocol

OpenFlow is a group of instructions that the controller used to manage the data plane elements. OpenFlow protocol supported the different types of messages to administer the switch. All these messages classified into three categories.

1. **Controller to Switch messages:**
   These messages are usually initiated by the controller and used to inspect the switch state (e.g.get information from the switch).

   - Packet_out: This is used by the controller to inject the switch by new flow entries or to a

forward packet(s) received by another switch. A message should include a list of actions that must be applied.

- Modify State: The controller uses this message to add, delete, or modify flows in the flow tables of the switch.
- Send packet: Used when the controller needs to send packets out of a particular port on the switch.
- Barrier Request/ Replies: Used for notifying the controller of completed operations from the switch.
- Features: This is the message exchanged between the controller and the switch after a secure connection is established. The switch replies to the controller by the same message with the capabilities that it can support.
- Configuration: This is an OpenFlow message that the controller used to setup configuration parameters inside the switch.

2. **Asynchronous messages:**

   These messages express the way the switch can talk to the controller without any permission to speak. This is where the switch can inform the controller of dropped packets or an interface going down.

   - Port status: Any change in the status of the port is sent to the controller; this includes a port going down.
   - Packet_in: When a switch receives a packet, and the packet does not match any flow entries, the switch sends this message to the controller.
   - Flow removed: This message is sent to the controller if any flow entry is removed from the flow table since any flow entry added is associated with an idle timeout, a hard timeout.
   - Error: This message is used by the switch to inform the controller of any errors it sees.

3. **Symmetric messages:**

   These messages are bi-directional and are used by the controller or the switch without any permission. For example, hello or echo requests are exchangeable messages between the sender and the receiver and can be replied without solicitation. There are three different types of these messages:

   - Hello: Like traditional network protocols, hello messages are exchanged between the controller and a switch on the startup stage.

- Echo: This message must be replied if any side receives one. The controller and the switch can initiate these types of messages, which can be used to give an idea about the latency of the switch to controller connection.
- Vendor: Any additional functionality that the switch offers can be sent to the controller by this message.

2.3 Single versus Multiple Controller Scheme

A controller is responsible for functions such as routing by maintaining the forwarding tables of the switches, whereas the switches in the data plane primarily perform packet forwarding functions. When a new routing path is required at the data plane layer, the switches must consult with their assigned controller for the routing decisions that the switches must make. In growing SDN networks, the deployment of multiple controllers is required to overcome the bottleneck that occurs when using a single physical controller. The structure of the SDN controllers is divided into two types as indicated in Figure 5.

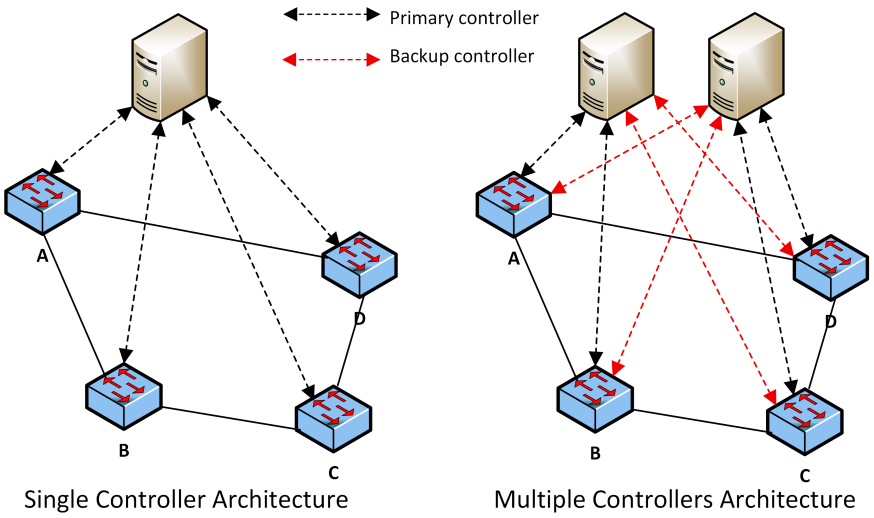

Figure 5: Single vs. Multiple Controllers architectures

- **Single Controller:**

  In this model, only one controller is administrating the whole network. In the early stages of the SDN appearance, the main tendency was to have a physically centralized controller to control the network. Therefore, the single controller can represent a single point of failure SPoF and affect the scalability aspects.

- **Multiple Controllers:**

  Unlike the single model, the physically distributed control layout, multiple controllers teams up to manage the network by deciding how packets should be forwarded by switches to achieve some level of performance and scalability. Moreover, multiple distributed controllers can assist each other in addressing any failure that could occur at any controller by exchanging their roles (primary/backup). HyperFlow [11] and Onix [12] are examples of SDN multiple controller schemes.

In the research we focus on multiple controllers architecture, and the structure of the multiple SDN controllers can be classfied into two types:

2.3.1 Centralized Control Scheme

In the centralized network, the controllers are logically centralized, physically distributed. All controllers are equally responsible for share and synchronize all information about the state of the whole network, and the decision is building based on the global network view. Furthermore, in a centralized scheme, it easy to ensure that the network is in a consistent, optimal configuration. On the other hand, in any newly established flows, an added latency becomes a bottleneck in large scale deployments. Additionally, as the scale grows, all advanced services are handled centrally, instead of locally.

2.3.2 Distributed Control Scheme

Differently, in a distributed design, all controllers will be portioned physically and logically. Each of which is in charge of managing a particular domain and has just a view of the domain it is responsible for. This feature can significantly enhance the scalability of the network. Practically, each distributed controller makes a decision of its managed domain. Other advantages can be noted in the distributed pattern, such as achieving high-availability and better latency during the handling of $Packet\_in$.

- **Flat Architecture**

  In the flat architecture, the controllers are positioned on the same level; each one manages a part of the network (see Figure 6). In other words, all controllers share the network view, and its responsibilities are equally divided.

- **Hierarchical Architecture**

  The hierarchical architecture is having three layers, as illustrated in Figure 7, where each layer contains a set of controllers. The top layer includes the root (Super) controller, which

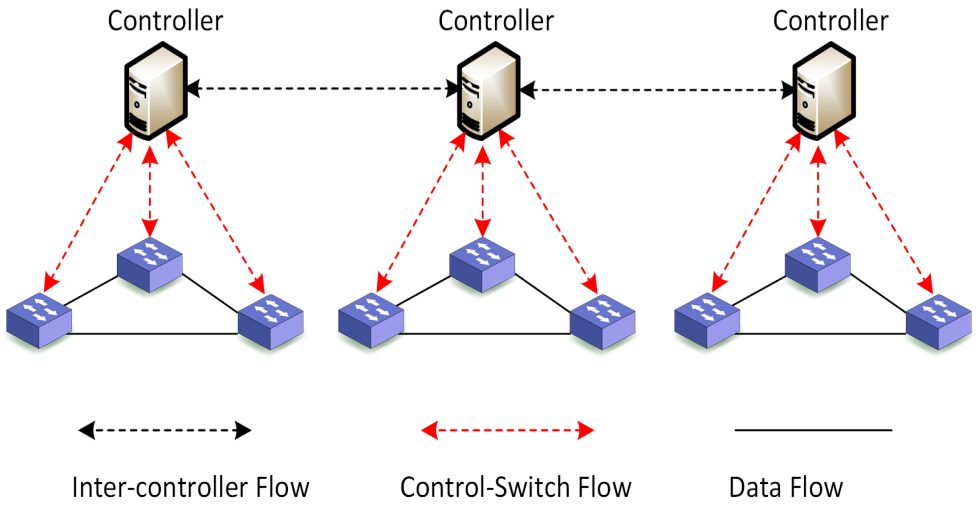

Figure 6: Flat Controller Architecture

administrates the domains' controller and synchronizes the global abstracted network view through a distributed protocol. A set of controllers are positioned in the middle layer, each one taking charge of managing its domain. The bottom layer includes all forwarding components of the data plane [13].

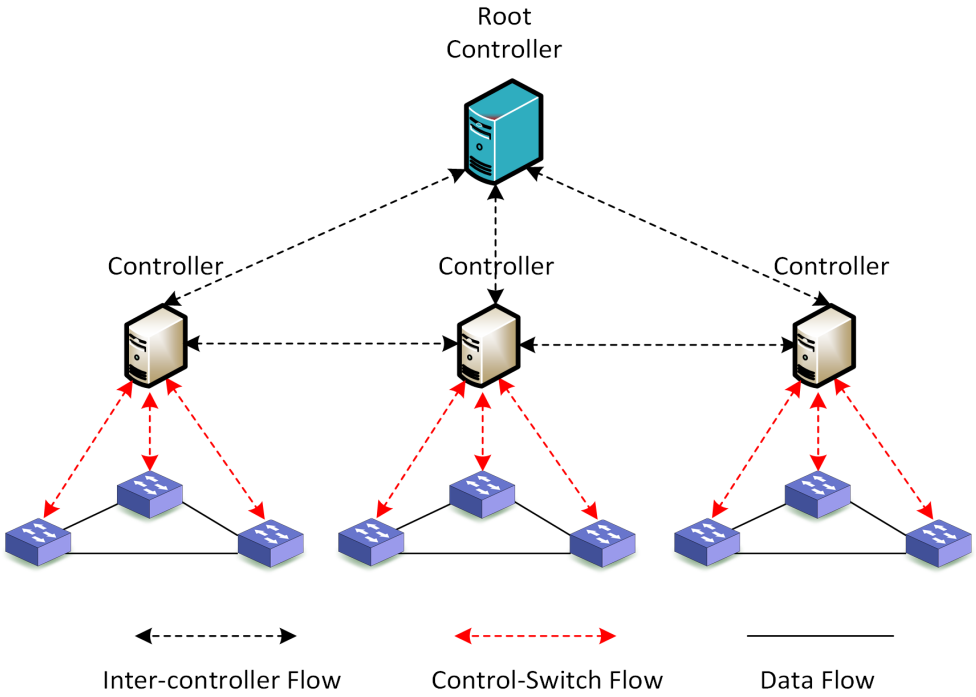

Figure 7: Hierarchical Controller Architecture

2.4 In-Band vs. Out-of-Band Control Scheme

The control connection concerning the data plane can be performed in two different ways: either by an In-band or Out-of-band connectivity, as illustrated in Figure 8.

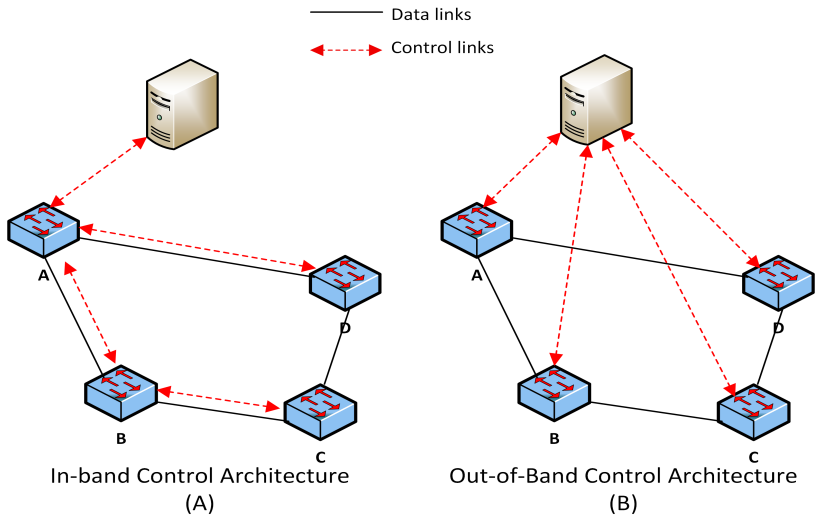

Figure 8: In-band versus Out-of-band Controller Architectures.

- **In-band Control**

  As shown in the Figure 8A, in the in-band approach, all control messages are exchanged through the same data plane channels. Reducing the cost of maintaining the network is one of the advantages of the in-band approach compared to the out-of-band scheme since separate control network channels are not required. Second, the in-band control does not require a separate port on a switch for the control channel. In-band also has drawbacks, for instance, the switch who is in charge of exchanging the control traffic with the controller (where the controller is placed) is at the risk of attack or failure and thus its role in redirecting control flows to the controller will end unless there is a backup alternative.

- **Out-of-band Control**

  Unlike in-band, Figure 8B demonstrates out-of-band where a dedicated network channel carries control packets. Generally, the control network architecture is constructed by dedicated physical links between the controller and every switch associated with that controller. The common example of using this approach is the data centers that are limited in geographical size. This pattern of networks design is considered costly to build, especially for small and medium organizations because of the requirement of separate networks [14]. Also, building a separate network may not be feasible in some scenarios (e.g., widely distributed central

offices in the access network). Besides, it has other major disadvantages. First, because of the separation of the control traffic from the data traffic. This raises some security concerns that are out of our scope. Second, failures of the control plane will also affect the data plane. Thus, a failure will disconnect the switch from its control component and make the network's recovery more challenging. For instance, the work proposed by Sharma *et al.* [15] and Guo and Bhattacharya [16] adopted the in-band scheme in the connection between controller and switches as an assumption to define a controller placement

Chapter 3

Combined Degree-Based with Independent Dominating Set Approach for Controller Placement Problem in Software Defined Networking

The main concept of SDN relies on abstracting the controller from the data layer to reduce the overhead of control messages between the control and data planes. Therefore, in the controller placement strategy design, which typically refers to how to optimally determine the location of controllers in the network and the association relationship between controllers and switches. It should be taking into account that the placement of the controller directly affects the latency between the controllers and switches, hence, which affecting the performance of the entire network. In this chapter, we aim to study the controller placement problem in SDN networks, design a strategy that relies on using the CDS concept. Following [17] in terms of placing controllers at the locations of the nodes, in this chapter, we provide an algorithm to dynamically determine which forwarding nodes are to be the primary host locations for the controllers. Furthermore, we present a method to guarantee the distribution of the controllers in the network. Compared to previous reference work, we significantly achieved an average low response time between the controllers and their associated forwarding nodes while also maintaining a low maximal response time between the nodes.

3.1 Introduction

The main idea behind SDN is to make the next generation of networks more dynamic and adaptive through the easy and rapid management of changeable network requirements. SDN is suitable for a network where each host is connected and managed by a single administrator, e.g., a campus, an enterprise, or a data center network. Conversely, it is hard to make traditional networks dynamic and adaptive because the network control software is implemented directly onto the network devices. In the latter, the network administrators must ask the network device vendors to implement new control mechanisms to meet their needs whenever a new network service or configuration automation mechanism is introduced into their networks. The key idea of SDN is to separate the network control logic (control plane) from the packet forwarding logic (data plane). A controller is responsible for functions such as routing by maintaining the forwarding tables of the switches, whereas the switches in the data plane primarily perform packet forwarding functions. When a new routing path is required at the data plane layer, the switches must consult with their assigned controller for the routing decisions that the switches must make. In growing SDN networks, the deployment of multiple controllers is required to overcome the bottleneck that occurs when using a single physical controller. The single centralized controller cannot meet the high demands of flow processing, especially when there is no match for the incoming packets in the existing flow entries at the switches [3]. Furthermore, a single controller has many limitations in terms of scalability, resilience, security, etc. [18]. Thus, a multi-controller environment is required to manage large scale SDN networks. However, deployment of multiple controllers requires a state synchronization between the controllers in order to maintain a consistent view of the network [19].

To make the controller placement problem distributed and scalable, controllers need to communicate with their switches such that routing updates, controller allocation information, etc., can be shared with them in a timely manner. Dealing with this problem can be simplified by considering two questions: how many controllers are needed to manage the whole network, and where should they be placed in the network? The best placement of controllers can be achieved in terms of different metrics and considerations based on the network's objectives. For this reason, getting the best possible placement requires studying new and efficient methods and metrics to determine the impact of the placement of controllers on SDN performance.

The objective of our controller placement approach is to minimize the response time between the controller(s) and the forwarding nodes to enhance the network's performance while considering different factors, such as the topology of the network and the shortest paths between the nodes.

3.2 Related Work

Many new networking problems have been raised since the emergence of SDN, particularly as such networks grow in size. One of the most debated and interesting issues is the SDN controller placement problem. Understanding the placement and number of required controllers is a key factor in answering SDN performance and fault tolerance questions. This design choice is called the Controller Placement Problem [3]. In this section, we review some previous works which formulate/solve the controller placement problem using various approaches.

Some research still emphasizes using Integer Linear Programming (ILP) formulation to find the minimum number of required controllers. Killi *et al.* [17] proposed a Mixed Integer Linear Programming (MILP) model called Controller Placement With Planning for failures (CPWP) in SDN to handle the controller placement problem. Due to the computationally intensive approach of ILP, this model is limited to small and medium-size networks. Later, to alleviate these limitations, the authors developed a simulated annealing heuristic to solve the problem of capacitated controller placement in large-scale networks taking into consideration controller failure(s) scenarios [20]. Similarly, Yao *et al.* [21] introduced another approach using ILP to solve the controller placement problem. This approach suffers from shortcomings, as it emphasized the importance of the delay time and the dynamic traffic load between the controller and switches and neglected the average delay time of all switches, which resulted in some switches being placed far from their assigned controllers.

Partitioning methods have also been discussed using different techniques. For instance, Xiao *et al.* [22] addressed the controller placement problem in WAN networks by partitioning the WAN networks into small subnetworks using spectral clustering method and then placing a controller in each small subnetwork. Zhong*et al.* [23] defined an algorithm called Min-Cover that aims to cluster the forwarding devices and then place one controller at each cluster's center. In their turn, Hu *et al.* [24] deal with controller placement while considering reliability. Their objective is to minimize the percentage of control path loss, but this model is considered effective only when the number of controller placements is at most 5. Li *et al.* [25] developed a control-domain adjustment algorithm calld (CDAA) based on Breadth-first search (BFS) method, where BFS is used to select the migration switches based on their distance to the its master controller and current traffic load.

Rath *et al.* [26] developed a game theory-based method to determine the appropriate locations of the controllers. Controllers can be dynamically added or deleted depending on their load. The key feature of this technique is improving the QoS and minimizing the controllers' deployment cost while not mentioning the specific controller placements. Sallahi *et al.* [27] introduced a mathematical formulation model to optimize the number of controllers by activating or deactivating the controllers and links to improve the network performance. Huang *et al.* [28] introduced a technique to

address the controller placement problem by developing a mathematical model aiming to optimize control plane utilization while simultaneously guaranteeing low network response time. The authors combined a scheduling algorithm named a gradient-descent-based (GD) to balance the trade-off between scheduling performance and problem scalability.

In a different trend, heuristic algorithms have been proposed. Hock *et al.* [29] optimize the placement of controllers in the context of reliability by developing a framework called Pareto Optimal resilient COntroller placement in SDN-based core networks (POCO). POCO achieves load balancing between the controllers and the switches. An extended POCO framework with heuristics has also been proposed by Lange *et al.* [30], where they considered only the latency between the switches and each controller to estimate how many controllers need to be deployed and their corresponding locations.

Wang *et al.* [4] have proposed an algorithm based on using optimized k-means to minimize the latency between the controller and its switches in a sub-network. Their experimental results showed that the maximum latency they get is shorter than one achieved by using the regular k-means algorithm. However, the authors adopted a distribution of the controllers' placements in terms of the farthest distance method between the current controller and the following selected one. This strategy suffers from some drawbacks, as a controller may be deployed on a node where there is only one link that could negatively affect failure and recovery issues. Moreover, the hierarchical architecture of multiple controllers [13, 31] is designed to solve the single point of failure problem in a large scale network. The implications of the multiple points of failure can be observed on their neighborhood links/nodes, where the abrupt increase in the network's load around the neighborhood of failures can overload on a certain number of nodes and links. Network Clustering Particle Swarm Optimization Algorithm (NCPSO) is a meta-heuristic optimization approach inspired by a natural process presented by Liu *et al.* [32]. The researchers utilized an evolutionary algorithm called particle swarm optimization (PSO) to address the controller placement problem. Loads of controllers were taken into account, which is the critical factor for large-sized networks. Singh *et al.* [33] conducted a comprehensive survey of the controller placement problem with analyzing the existing solutions and the constraints associated with it.

3.2.1 Preliminary concepts

The challenges generated by SDN networks require applying diverse mechanisms until reaching a suitable solution. Therefore, in this section, we introduce some preliminary concepts of the techniques we use in this chapter.

Controller-Switch Response Time

An early attempt to define the controller placement in SDN was by Heller *et al.* [3], where they formulated the controller placement problem as an optimization problem by considering different criteria, which include average response time and maximum response time. These two metrics are defined as follows:

- **Average Response Time (RS_{avg})**

  The objective of this metric is to determine a controller placement that will minimize the overall average response time from the controller to every node. It corresponds to the minimized p-means optimization problem. The goal is to find the p centers that minimize the sum of the distances between a set of nodes and the p centers [4].

$$RS_{avg} = \frac{1}{n} \sum_{v=V} \min_{c \in C} \; d(v, c) \tag{1}$$

- **Maximum Response Time (RS_{max})**

  The goal of this metric is to find a controller placement that minimizes the maximum response time from the controller to any forwarding node. This corresponds to the minimum p-center optimization problem that aims to partition a set of nodes into p clusters. A specific node is attached to the cluster with the closest mean.

$$RS_{max} = \min_{v=V} \max_{c \in C} \; d(v, c) \tag{2}$$

High Degree (HD) Clustering Algorithm

The High Degree clustering technique is a connection-based clustering approach. The degree of a node is the number of connected links from other (neighboring) nodes to the node. A node with the highest degree in the network is chosen as a cluster center [34]. As indicated in Figure 9, node 6 is a high degree node with 5 links.

Dominating Set (DS)

Dominating sets were studied as early as 1862 when de Jaenisch [35] studied the problems of finding the minimum number of queens that are necessary to cover or dominate an $n \times n$ chessboard. Berge [36] and Ore [37] in 1962 formalized this subject in graph theory.

A dominating set is a subset S of the nodes of a graph G such that every node in G is either in S or a neighbor to a node in S. Dominating sets are widely used in networks clustering where it can be classified into three major types; 1) Independent Dominating Sets (IDS), 2) Weakly Connected Dominating Sets (WCDS), and 3) Connected Dominating Sets (CDS) [1].

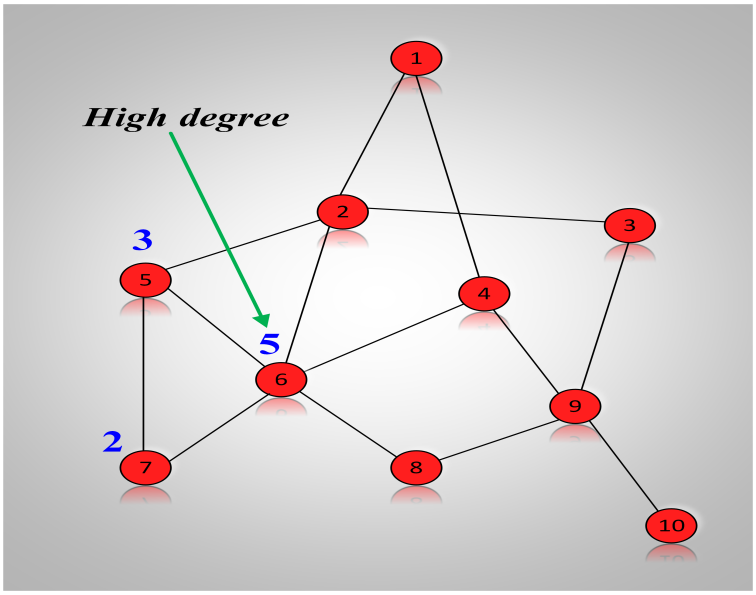

Figure 9: High degree node concept

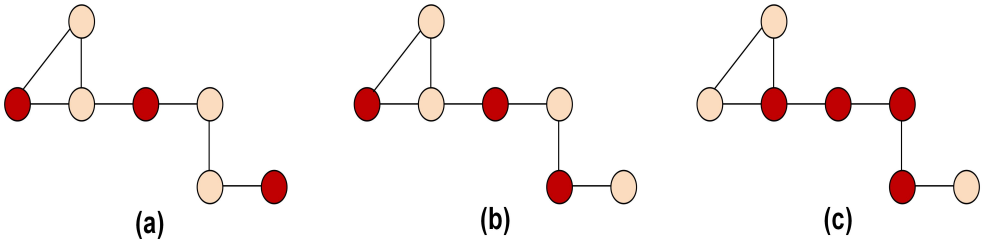

Figure 10: (a):IDS, (b):WCDS, (c):CDS;
Based on [1]

- **Independent Dominating Sets (IDS)**

  IDS is a dominating set S of a graph G in which there are no adjacent vertices in S. Figure 10.a shows an example IDS where the red nodes indicate the cluster centers.

- **Weakly Connected Dominating Sets (WCDS)**

  S is a subgraph set of graph G, a weakly induced subgraph $(S)w$ consists of all vertices of S, its adjacent, and all links with graph G with minimum one endpoint. A subset S is a WCDS if S is dominating and $(S)w$ is connected. Figure 10.b show a WCDS example.

- **Connected Dominating Sets (CDS)**

  A connected dominating set (CDS) is a subset S of a graph G such that S forms a dominating set and S is connected. Figure 10.c shows a sample CDS.

The idea is to find a minimum dominators problem that is formulated for the context of a network

where the objective is to find a minimum number of controllers that allow all the other forwarding nodes to be controlled by at least one of the selected controllers. However, finding a minimum dominating set is NP-hard, which means there is likely no polynomial-time algorithm can guarantee an optimal solution. Therefore, we will consider a heuristic approach.

3.3 Problem Formulation

We represent the network as a graph $G(V, E)$ where V represents the node set in the network topology, and E represents the link set, the connections between pairs of distinct nodes in V, the weight of which is the propagation delay of the link. In SDN, the nodes and controllers are the forwarding elements; therefore, we make the distinction in our work of assuming that the controller locations are at node locations in the network. Let $C = (c_1, c_2, ..., c_n)$ be the set of controllers to be deployed and let $S = (s_1, s_2, ..., s_n)$ denotes the set of switches in the network such that $V = C \cup S$. When a node is selected as a controller, it is moved from S to C. Also, we define $d(s, c)$ as the shortest path distance from a node $s \in V$ to $c \in V$ (mainly we are interested in the shortest distance from a forwarding node $s \in S$ to its associated controller $c \in C$).

Determining the controller placement is an optimization problem where we can identify a solution by optimizing an evaluation metric. In this part, we use response time as the evaluation metric. Traditionally, response time represents the time taken by a packet to travel from the source node to the destination node. At the same time, in the SDN approach, we will assume the response time is the time taken from the forwarding node to the controller and that this time is proportional to the distance between them. The links between nodes in the topologies we will consider for our simulations are known to be fiber optics, for which the signal propagation is nearly at the speed of light. To maintain authenticity, the propagation delay for a link was calculated by dividing the distances between nodes by the speed of the light. The main goal in this chapter is to specify a required number of SDN controllers and to determine their placement and associated assignment of the controllers to the forwarding nodes in the data plane, such that the response time is minimized. Generally, our main objective is to minimize controller to node delay time in terms of average and maximum values.

3.4 Proposed Approach

In this section, we walk through in detail of how the algorithm works, supported by an example. We propose to use a network partition using a high degree node approach to specify the placement of the controllers needed in the network.

3.4.1 High Degree with Independent Dominating Set HDIDS

Unlike previous works, our approach relies on defining the candidate nodes for controller locations to ensure minimum response time between the controller and its associated switches and minimizing the maximal response time as well. We look at the overall network distribution in terms of obtaining a minimum response time. Towards this end, our algorithm consists of two phases to deal with controller placement: (1) iteratively determining candidate controller instances from S using a high node degree method until all nodes in S are dominated by nodes in C, or the maximum number of controllers is reached (details of this iterative algorithm given in Algorithm 1), and (2) partitioning the network into multi-domains exploiting the independent dominating set approach to ensure controller distribution in a manner that achieves as close as possible the minimum response time. The aim of using this strategy is to determine high degree nodes that represent cluster centers, in order to build the network subsets for forwarding nodes at the end. The major selection factor of this approach builds on iteratively selecting the node with a high degree and minimum total distances to other nodes in the network.

Here we give a detailed description of the controller selection algorithm. With a given number of controllers, k, to be deployed in the network, in our proposed algorithm, we iteratively aim to detect the near-optimal location s from the set of potential locations set S not sharing an edge with any of the selected controllers in C (i.e., s is not dominated by any node in C).

Algorithm 1 Controller Selection

Require: $G(V, E)$

1: $S \leftarrow V$, $C \leftarrow \emptyset$
2: **while** $C \neq \emptyset$ **do**
3: $S' \leftarrow \textsc{NodesWithMaximumDegree}(S, G)$
4: **if** $|S'| = 1$ **then**
5: $C \leftarrow C \cup (S' = \{s\})$
6: **else**
7: $s \leftarrow \textsc{NodeWithMinTotalShortestPaths}(S', G)$
8: $C \leftarrow C \cup \{s\}$
9: **end if**
10: $S \leftarrow S \setminus (\{s\} \cup \textsc{NeighborNodes}(s, G))$
11: **end while**
12: **return** C

Initially, C is the empty set and $S = V$. At each iteration, the nodes in S that are not dominated by any node in C, call this set S', are considered. All the nodes with a maximal degree in S' are selected. If there is only one node with a maximal degree in S', then this node s is selected. If there is more than one node with a maximal degree in S', then out of this subset of maximal degree nodes, the node s is selected that has the smallest total of the shortest path lengths to all other nodes in the network. For calculating the shortest paths among all nodes in the network, the path finding algorithm A* algorithm with a Haversine heuristic [38] has been employed. Then the selected node s is removed from S and added to the controller set C. The iterations end when all the nodes (if any) remaining in S are dominated by nodes in C, or the number of nodes in C is k. In Algorithm 1, we reference some simple procedures:

- **AllNodesWithMaxDegree**(S, G)**:** returns a set of all the nodes with the maximum degree from the set S.
- **NodeMinTotalShortestPaths**(S', G)**:** returns a node with the smallest total of shortest paths to all other nodes in G.
- **NeighborNodes**$(s, G, S))$**:** returns the set of all neighboring nodes of s which are in S.

In all cases, where more than one node meets the node requirements, one node is chosen randomly.

3.4.2 Example: Clustering Description by HDIDS Algorithm

In this example, we use the topology shown in Figures 11a to 11d to illustrate the mechanism steps of HDIDS algorithm for finding the best number of the controllers and their placements.

The first step of HDIDS algorithm is to find a node with a maximum connection degree to select it as the first controller. As shown in the Figure 11a, node (5) has degree $d = 5$ (we denote the degree of node connection with the symbol d); hence, node 5 is selected as the first controller. In the case of multiple nodes with the same degree, we choose the node that provides a minimum total of the shortest paths lengths to all other nodes in the network.

The next step is to find the potential location of the second controller. This can be achieved by selecting the second node that meets some conditions: (1) if exists, the node with the same connection degree of the first controller or the next level of connection degree with the minimum total distances to other nodes, and (2) it does not have a direct connection with the first controller.

As shown in Figure 11b, there is no node with the same connection degree as node 5. Therefore, node 8 is the second controller with a connection degree $d = 4$. Each node will be assigned to the nearest controller using the shorter paths.

Similarly, for the third controller, it can be seen that there is more than one node with degree 3 (1, 3, 4, 6, 7, 9, and 11). In this instance, nodes 1, 3, 4, 6, and 7 have a direct connection with

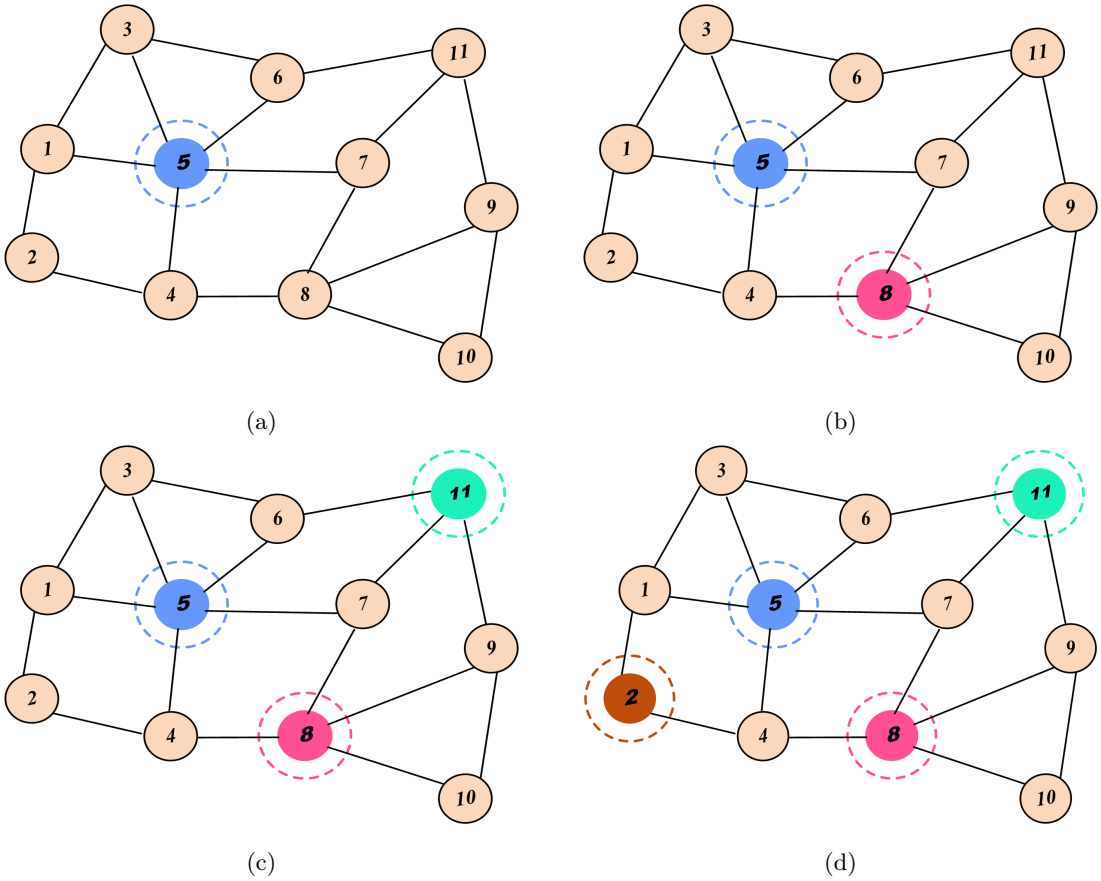

Figure 11: Controller Selection Scheme

the first controller (node 5), while nodes 7 and 9 have a direct connection with the second controller (node 8). Hence, as the only node of degree 3 with no direct link to a chosen controller, node 11 is selected as the third controller as illustrated in Figure 11c.

One of the most important features of our approach is that the maximum number of controllers is determined according to the structure of the network and the communication between the nodes. As apparent in Figure 11d, the only node that topology does not have any direct connection with any of the previously selected controllers is node 2. Hence, node 2 is selected as the last possible controller placement.

As a result, we end with 4 controllers as the optimum number of controllers with their placements. The domains after assigning each node to its appropriate controller (with the shortest path from that node to any controller) are shown in Figure 12.

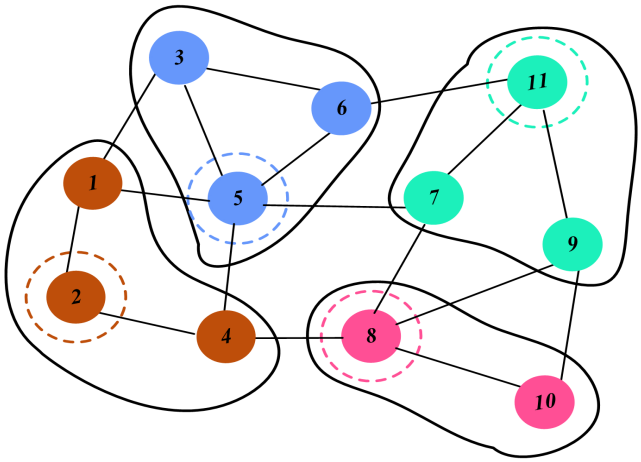

Figure 12: Final controller selection scheme

3.4.3 Time Complexity of HDIDS algorithm

In this section, we highlight the time complexity of the HDIDS algorithm.

First, AllNodesWithMaxDegree(S, G) procedure can take up to $O(n^2)$ time. A graph can have up to $n(n-1)/2$ edges. To calculate the degrees of all the nodes, we need to look at all these edges (i.e., count the number of neighbors of each node). Once all nodes' degrees calculated, then they will be stored in a max-heap. Building the heap, and maintaining the heap over the running of the algorithm would be $O(n \log n)$.

Second, For NodeMinTotalShortestPaths(S', G) procedure, it requires a maximum of n iterations over the entire running of the algorithm (assuming that S', G represents all the nodes in the worst case). Hence, for every node in S', G, the time complexity of this is will take $O(n^2 \log n)$ time.

3.5 Results and Discussion

All the experiments described in this section were carried out on an Intel(R) Core(TM) i7-6770 CPU @3.40GHz and 16GB RAM with Windows 7 Pro (64-bit) operation system.

In order to evaluate the performance of HDIDS, various evaluation experiments were conducted. We evaluate our approach on networks of different sizes and structural topologies adopted from Internet topology Zoo [39]. Note, since these are real topologies, the distance between the nodes represents a link weight. Table 5 shows the main characteristics of experimental topologies.

In the following part, we first compare the response time performance between HDIDS and Optimized K-means (OK-Means) algorithm by Wang *et al.* [4]. We evaluate the algorithms in two aspects, the first is the average response time, while the second aspect is the maximal response time

Network topology	Number of nodes (t)	Number of links
Internet2 OS3E	34	42
ATT North America	25	57
Bell Canada	48	65
TataNID	145	196

Table 1: Main characteristics of experimental topologies

of any node associated to its controller. And we further characterize both that average and maximal response time performance against the number of controllers. The results of the simulations are depicted in the following section.

3.5.1 Average Response Time

Figure 13 compares the number of controllers with their impact on the average response time on the Internet2 OS3E topology. We can observe that for all numbers of controllers up to 12, and our algorithm achieves better average response time than that achieved by Optimized K-means. However, the gap decreases with the number of controllers (with the increasing overlap in the selected sets of controllers). For example, for 10 controllers, our algorithm HDIDS achieves 1.83ms while Optimized K-means has an average response time of 2.49ms.

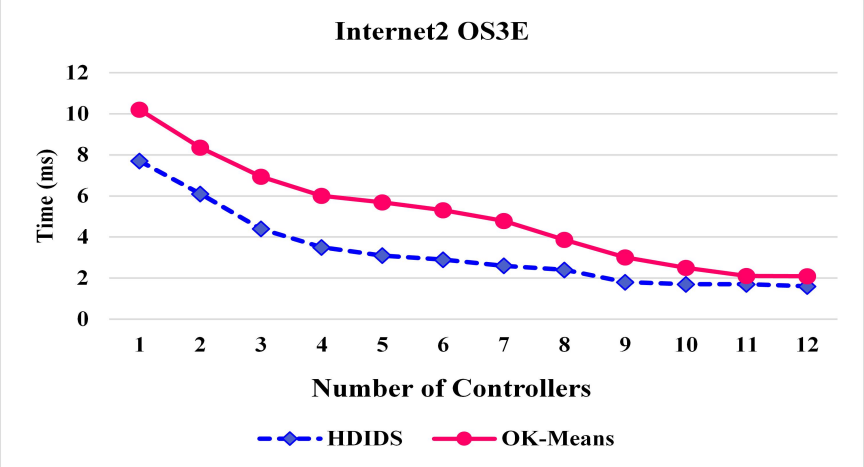

Figure 13: Average Response time in Internet2 OS3E

A similar pattern is repeated for the other three experimental topologies. For instance, for the Bell Canada topology as shown in Figure 14, for fewer than 8 controllers, HDIDS has substantially better performance in terms of average response time.

When the number of controllers increases past 8, though, Optimized K-means has slightly better

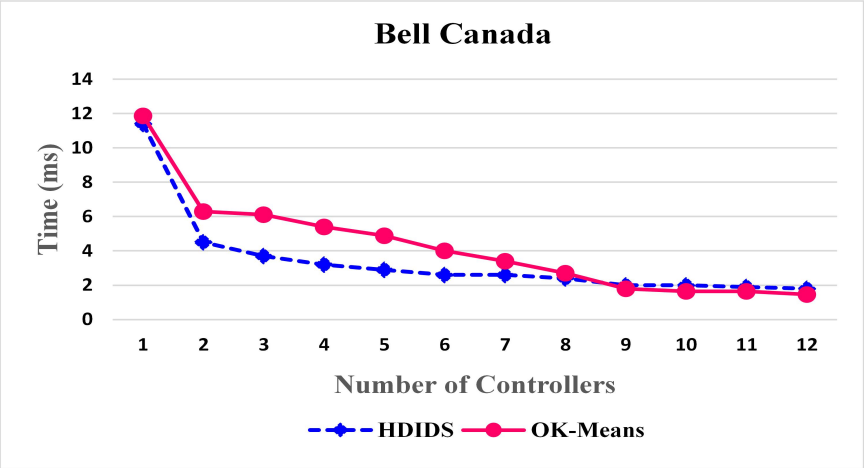

Figure 14: Average Response time in Bell Canada

performance. For example, for 10 controllers, HDIDS obtained 2.0ms while Optimized K-means obtained 1.65ms as average response time. Similarly, in the ATT topology Figure 15, we see that HDIDS has better performance until about 6 controllers where for more than 6 controllers Optimized K-means performs better. For 8 controllers, Optimized K-means achieved 1.5ms, which is less than the 1.7ms by our approach HDIDS. The featured contribution of our algorithm, is that it achieves a much better average response time than Optimized K-means in large networks. As a good side effect of SDN, it gives the network operator more scalability by having the ability to manage and change network infrastructure at any moment.

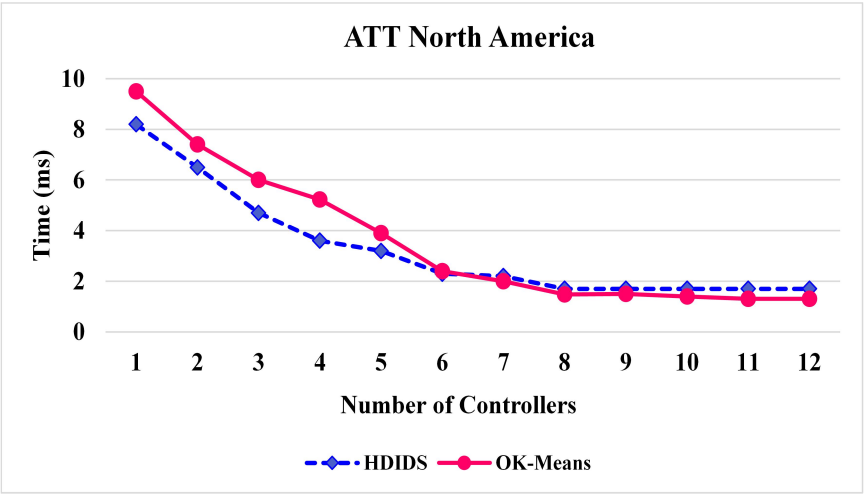

Figure 15: Average Response time in ATT North America

The above three networks (Internet2 OS3E, ATT North America, and Bell Canada) would be considered to be medium-sized networks (Heller *et al.* [3] where he defined a medium-size network as

having 18 to 24 nodes). Therefore, TataNID topology has been chosen as an example of large network with 145 nodes to show the effectiveness of our algorithm. Here, for all numbers of controllers, as seen in the following Figure, the average response time for HDIDS is significantly better than that achieved by Optimized K-means. For instance, when the number of controllers is 11, we can observe HDIDS gets 1.83ms while Optimized K-means achieves 3.79ms.

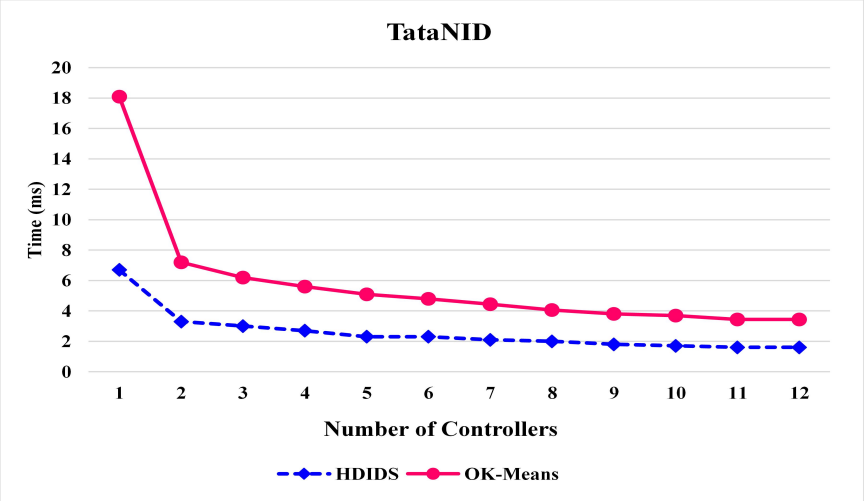

Figure 16: Average Response time in TataNID

Figure 16 also demonstrates the effectiveness of increasing the number of controllers for large networks. It can be seen that changing the number of SDN controllers from one to two results in a reduction of up to 50% overall in the average response time. Further increase in the number of controllers beyond 10, though, has a much less significant effect on average response time. Table 2 illustrates a selection of the number of controllers in the experimental topologies and their corresponding average response times.

Topology	Internet2 OS3E	ATT	Bell Canada	TataNID
HDIDS	k=10	k=8	k=9	k=10
	1.73ms	1.72ms	2.0ms	1.81ms
OK-Means	k=11	k=10	k=10	k=10
	2.11ms	1.46ms	1.65ms	3.79ms

Table 2: Illustration of the number of controllers (k) and corresponding Average response time

If we were to consider an average response time threshold between controllers and switches that our SDN must remain below, Table 3 describes the corresponding number of controllers for both algorithms within average response time thresholds equal to 2ms, 3ms, and 4ms, respectively.

Threshold	2ms		3ms		4ms	
	HDIDS	OK-Means	HDIDS	OK-Means	HDIDS	OK-Mean
Internet2 OS3E	9	12	5	9	4	8
Bell Canada	9	9	5	8	3	6
ATT	7	7	5	5	4	5
TataNID	8	> 12	4	>12	2	10

Table 3: Minimum number of controllers required threshold of 2ms, 3ms and 4ms average response time

Our algorithm HDIDS requires 9 controllers in Internet2 OS3E and Bell Canada topologies for the average response time threshold of 2ms, while it requires 8 controllers in TataNID topology (as compared to more than 12 for Optimized K-Means). For threshold times from 2ms to 4ms, HDIDS reduces the number of required controllers between 0% to 50% compared to Optimized K-Means in medium-sized networks. While in TataNID topology as an example of a large topology, HDIDS decreases the controller number between 33% to 80%. Obviously, HDIDS requires fewer controllers than Optimized K-Means for all networks for the threshold of 2ms, 3ms, and 4ms except three cases where the number of controllers is the same.

3.5.2 Maximal Response Time

In this section, we illustrate the results obtained by our algorithm HDIDS in terms of the maximal response time. The findings presented in the following two figures show a reduction in the maximum response time in Internet2 OS3E and Bell Canada topologies, respectively. Since choosing the node with the minimum total distances to all other nodes in the network is an initial step in both Optimized K-means and our algorithm for finding the placement of the controllers, we can observe there is an match of the maximal response time in some topologies as indicated in Figure 17 and Figure 18. In Figure 17, it is clear that the maximal response time of Internet2 OS3E topology in both approaches for $k = 1$ is the same (15.4ms).

Considering Bell Canada topology as indicated in Figure 18, the maximal response time is (18.43ms) for the Bell Canada. Afterward, the maximal response time decreases with the starting partitioning process, with HDIDS consistently outperforming Optimized K-means. The maximum response time for both HDIDS and Optimized K-means approaches in Internet2 OS3E and Bell Canada topologies is reduced. For example, it decreased to 15.5ms to 3.4ms, and from 18.43ms to 6ms in Internet2 OS3E and Bell Canada topologies respectively when number of controllers is 4.

Interestingly, Optimized K-means as shown in Figure 19 has better performance in terms of

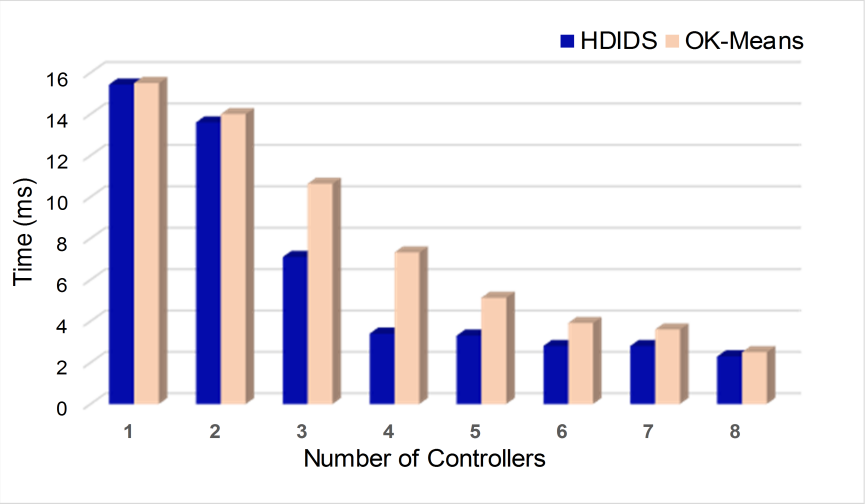

Figure 17: Maximal response time of Internet2 OS3E

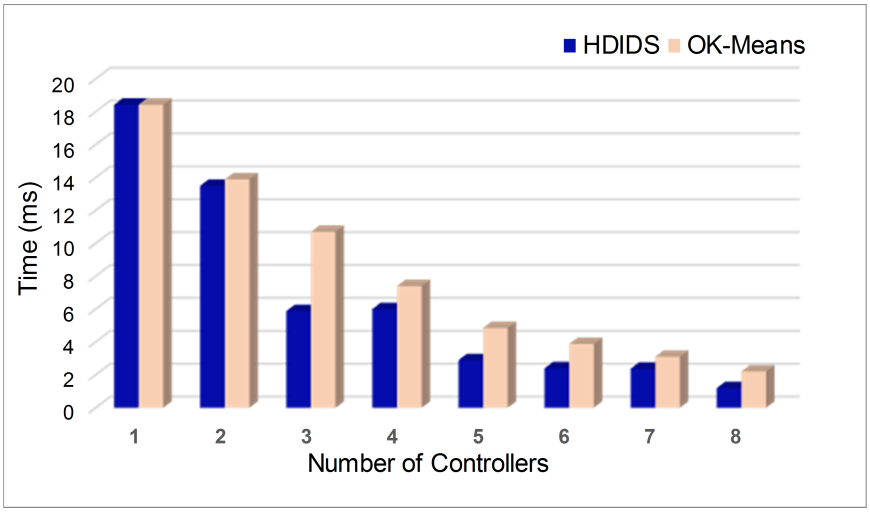

Figure 18: Maximal response time of Bell Canada.

maximal response time for 1 or 2 controllers and then again, although only slightly, when the number of controllers is 6 or more. As mentioned above, our algorithm produced a better performance in large network topologies and this also holds in terms of the maximal response time.

As appears in Figure 20, HDIDS decreased maximum response time from 13.9ms for 1 controller to 1.2ms for 9 controllers, with HDIDS consistently outperforming Optimized K-means.

Again, if we were to consider, in this case, a worst case response time threshold that our SDN must remain below, Table 4 illustrates the corresponding number of controllers in our algorithm HDIDS and Optimized K-Means algorithm within average of the minimum maximal response time threshold equal to 2ms, 3ms, and 4ms, respectively.

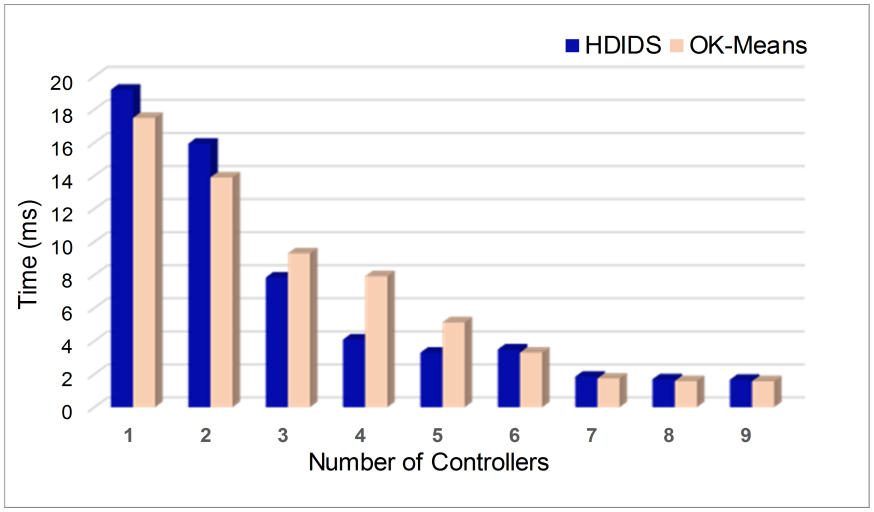

Figure 19: Maximal response time of ATT North America.

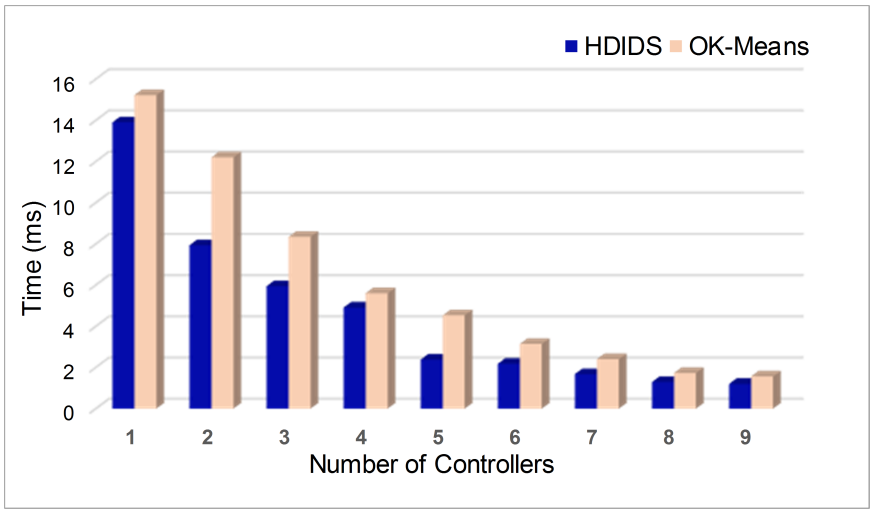

Figure 20: Maximal response time of TataNID.

Table 4 shows that for both Internet2 OS3E and ATT topologies the required number of controllers within a 2ms maximum response time threshold are the same. This is due to these topologies being located in the same geographical area. While in Bell Canada and TataNID topologies, our algorithm HDIDS achieves a number of controllers fewer than the Optimized K-Means algorithm. Also, HDIDS achieved a clear distinction in the number of controllers required within 3ms or 4ms maximum response time in all four topologies.

To our knowledge, the optimal number of controllers required for these experimental topologies for either average or worst case response times have not been presented in the literature. All the previous related work considered different parameters or metrics to evaluate the controllers' optimal

Threshold	2ms		3ms		4ms	
	HDIDS	OK-Means	HDIDS	OK-Means	HDIDS	OK-Mean
Internet2 OS3E	8	8	4	7	4	6
Bell Canada	6	8	5	5	5	6
ATT	7	7	5	6	4	6
TataNID	6	8	5	6	5	6

Table 4: Minimum number of controllers required threshold of 2ms, 3ms, and 4ms maximal response time

number and placement, so it could not be compared directly with our results. Similarly, some previous works conducted different experiments on random topologies to determine the optimal number and placement of controllers.

3.6 Conclusions

In large SDN networks, it is very hard to manage the entire network with one controller unit due to the increased number of forwarding nodes andthe increased number of routes that must be managed as well. Geographically, with larger SDNs, there is an increased lag between the forwarding nodes and their controller. As a result, the controller units are no longer able to provide optimal network performance or QoS. Hence, using a certain number of controllers to handle this issue is desirable. In this chapter, we propose a new technique named HDIDS to deal with the controller placement problem with regards to minimizing the average and worst response time between the controllers and their assigned forwarding nodes.

Our approach in HDIDS is based on selecting the available controller with the highest connection degree and then exploiting the independent dominating set method to automatically form clusters of forwarding nodes associated with each controller in the network. We conducted experiments under many different metrics and topologies sized from medium to large. Compared with previously published work, experimental results showed that our approach provides better results, especially in large networks, in minimizing response time and maximal response time reduction as well. The problem of SDN controller placement has several aspects that still need to be understood, where further investigation is required. For example, covering different performance metrics such as reliability and survivability.

Chapter 4

Survival Backup Strategy for Controller Placement Problem in Software Defined Networking

The controller placement is of paramount importance in the SDN scenario, as explained in the previous chapters. Besides that, in order to construct an effective and reliable SDN control network, in this chapter, we propose a backup controller scheme that finds the best locations for the backup controllers for fast recovery from the controller's failures. The main goal of this scheme is to enhance the survivability of the multiple controller architecture.

4.1 Introduction

The core objective of SDN is to provide a version of next generation networks that are more dynamic and adaptable in terms of management of changeable network requirements. A controller is in charge of the controlling functions, such as configuring the forwarding tables and setting up network paths of all switches. Complementary, the switches in the data plane primarily perform packet forwarding functions. When a new routing path is required at the data plane level, the switches must communicate with their assigned controller for the routing decisions that the switches must make. In large networks, one controller is not sufficient to manage network resources, particularly with respect to addressing the high demands of flows processing [3]. Furthermore, a central controller may subject to Single Point of Failure (SoPF) [18], where if there is a fault in the controller, it will lead to whole network performance deterioration. The SDN is being used for increasingly large networks in order to meet the scalability, reliability, and availability demands. However, in order to

maintain a consistent view of the network, a multiple controller model requires state synchronization between the controllers. Given longevity and reliability requirements, it is important to ensure the highly robust networks by enhancing the reliability and protection levels. The multiple controller environment brings convenience, however, it may suffer from some inevitable problems that lead to network failure (e.g., the possible failure of a controller itself). In the case of a controller's failure, the connected switches will lose the connections to their associated failed controller unless they can be migrated to an alternate controller. Nevertheless, without careful management of how these switch migrations occur, those alternate controllers are still vulnerable to failure due to exhaustion of their capacity. Therefore, a proper controller backup approach is necessary to recover a broken network from controller failure [40].

To our knowledge, the previous works on controller backup strategies were based on selecting one of the existing controllers as a backup controller that will replace the failed primary controller by migrating the affected switches to the backup controller. Assigning one of the existing controllers as a backup controller in case of controller failure may deem inappropriate due to the harm it may cause to this controller by overloading it with the new switches traffic, hence, in turn, increasing its failure probability. There are two assignment patterns from the perspective of using the existing controllers as a backup. On the one hand, some techniques re-assign *all* switches managed by the failed controller to the nearest controller, which requires administrative intervention [17]. On the other hand, other techniques reallocate each switch under the failed controller *separately* to the nearest controller for that switch. These techniques will need to reconstruct the whole network, such as updating the policies, routing, and flow tables for all controllers, which is a time-exhaustion task [41]. Survivability of the control network in case of controller failure recovery is a crucial concept to maintain the continuity of the network operations. Therefore, the survivability-based assignment is also a significant consideration to meet the Quality of Service (QoS) requirement in SDN networks.

Getting the best placement of the controllers in SDN networks requires further investigation to develop efficient methods and metrics to determine the impact of the controllers' placement on the performance of SDNs. Motivated by those as mentioned earlier, we aim to make the network controllers' structure robust enough to resist the failure by designing a dynamic, flexible backup approach that can make the network survivable even after controller failure event(s).

4.2 Related Work

In SDN, by decoupling the control plane from the data plane, all the control decisions are incorporated into a (logically) centralized entity called a controller. However, the controller placement problem (CPP) in SDN can be summarized as where to place controllers and how many to use [3].

In the following section, we go through some previous research that formulates/solve the controller failure problem in different trends.

4.2.1 Overview on SDN Survivability

Many researchers have considered the controller placement problem in recent years. However, other related aspects, such as resilience, availability, reliability, still need more investigations. In the following part, we review some works related to these aspects of the controller placement problem. Particularly, we present work on the backup controller placement in SDN.

Survivor is a representative work presented by Müller *et al.* [42]. It is a controller placement technique aiming to improve the connectivity, capacity, and recovery as a survivability aspect. Two different methods are introduced by the authors; (i) an integer linear programming model for enhancing the connectivity between the switches and controllers by increasing paths diversity, while guaranteeing that the controller capacity will not be exceeded; (ii) improving fault-tolerant by composing a controllers' backup list to increase recovery efficiency.

In [43], Zhang *et al.* treat the backup controller placement problem by setup a backup controller for each primary domain to handle connectivity. However, the authors did not consider the total placed number of (primary/backup) controllers and the communication overhead issues that will arise in the network. Botelho *et al.* [44] focused on performance aspects of the control level by addressed the consistency between a network controller and their set of backups. The authors conclude that acceptable performance can be achieved with strict consistency and fault-tolerant systems. Although Vizarreta *et al.* [45] did not take the controllers' capacity into consideration, they proposed two resilient strategies that provided the shortest working and backup control paths for dealing with link and node failures. The first method involved joining each switch to its assigned controller through two disjoint paths, while the second method connects each switch with two controller instances via disjoint paths. The authors evaluate their solution's performance using the expected control path loss and the average control path availability. Yang *et al.* [46] addressed the controller placement problem considering single and multi link failure. The authors developed a greedy algorithm to place the controllers greedily and iteratively based on the link failure rate and applying Monte Carlo simulation to reduce the computational overhead.

For QoS and resilience considerations, Tanha *et al.* [47] proposes two heuristic algorithms based on graph theory technique to solve the backup problem. The authors take into account both the latency between the switches and the controllers and the capacity of the controllers. Furthermore, a polynomial-time algorithm proposed to solve this NP-hard problem.

An optimization model for minimizing the cost of the network with backup capacity suggested by Killi and Rao [48]. The model enhanced the mapping between the switches and the controllers

while achieving full resilience against pre-specified controllers' failure. Although this approach uses the existing core controllers as backup instances with using a reserved capacity on the controllers, it may suffer the consequences of cascading failure of the controllers. According to [49], Hu *et al.* proposed a reliable and load balance-aware multi-controller deployment (RLMD) scheme to assure balancing the load rate between the controllers and the switches within the multiple clusters. Their approach efficiently managed to balance the allocation of the controller traffic loads but is suffer addressing the fault tolerance issue.

4.2.2 Improving SDN Controllers survivability

In the distributed multiple controller SDN architecture, backup controllers will be instantly assigned to the switches previously managed by the failed controller. However, one controller's failure can significantly affect the backup controller due to the added load (overload). In the worst case scenario, generating consecutive failures to other controllers leads to the crash of the entire network. This phenomenon is known as controllers cascading failure [50]. Figure 21 illustrates a simple example of the controller cascading failure phenomenon.

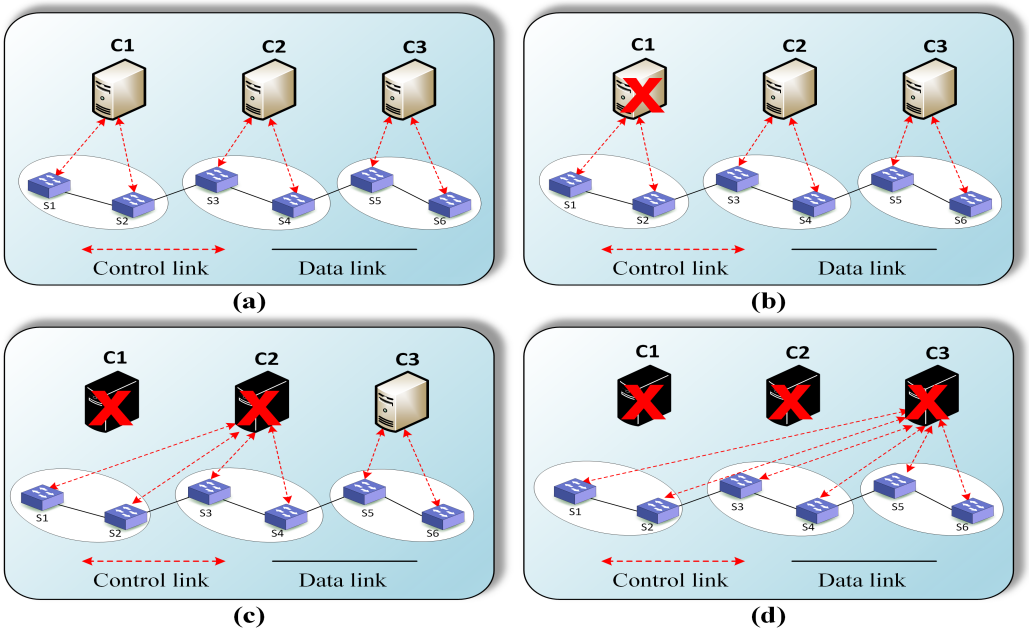

Figure 21: The controller cascading failure phenomenon

As shown in the example, an SDN network with three controllers C1, C2, and C3, and each controller manages two switches. As a controller C1 fails, as shown in Figure 1(b), its associated switches will be disconnected. According to the existing backup controller methods, the disconnected switches of C1 are migrated to the nearest controller, which is C2. Unfortunately, due to the

limitation of the controller's capacity, C2 cannot simultaneously manage the switches (S1 and S2) of the failed controller C1 with its switches (S3 and S4). The capacity of controller C2 will be exhausted by the flow requests of (S1 to S4), as a result, C2 will fail. Likewise, the disconnected switches S1 to S4 will be reassigned to the controller C3, and for the same reason, C3 also fails. Finally, all the controllers in the network have failed, and the entire network collapses.

A controller's capacity is a factor to consider in maintaining the stability of network performance, where overloading the controller would generate more severe consequences on the network. In the above example, we can notice that an existing primary controller's assignment as a backup controller is an inadequate choice, especially if the controllers' capacities are almost fully utilized. Therefore, it is necessary to introduce a dynamic backup controller assignment mechanism to increase the control plane flexibility in order to deal with the failure scenarios.

4.3 Network Model and Problem Formulation

The network modelled as a connected graph $G(V, E)$ where V represents the node set in the network topology, and E denotes the set of links. The connections between pairs of distinct nodes in V, the weight of which is the propagation delay of the link based on their geographical locations. In SDN, the controllers can be placed at the same location of the switches. Therefore, we assume the primary controllers are placed in the locations of switches (i.e., $C = S$). Let $C = (c_1, c_2, \ldots, c_n)$ be the set of controllers to be deployed and let $S = (s_1, s_2, \ldots, s_n))$ denotes the set of switches in the network such that $C \subseteq S = V$. The distance $d(s, c)$ defined as the shortest path from a node $s \in V$ to $c \in V$. Further, the shortest path computed by using A* algorithm associated with a Haversine heuristic [38]. The desirable goal is to achieve a fault-tolerant control plane through improving the backup controller approach under the controller's failure scene. Generally, our main objective is to find the best placements of the candidate backup controllers inside each VBD that satisfied the minimum response time in terms of average and maximum values.

4.4 Proposed Approaches For Survivable Backup Controller

4.4.1 Our Assumption

Virtual Backup Domain (VBD)

To further enhance the control services survivability of the network in case of a controller failure, we present a novel survivable approach to be implemented on top of the primary controller selection approach. The core component of our strategy is the construction of what we term a *Virtual Backup*

Domain (VBD). A VBD is a grouping set of one or more domains associated with the primary controllers (here we consider our approach HDIDS selection) in the network, facilitating the selection of the backup controllers. The main idea of this method is to combine k of the domains (typically neighbouring domains) produced by the primary controllers' selection approach to generate a new VBD. If we assume that there are n such domains, then grouping (packing) k of them together into one VBD such that $k < n$, hence, at least $\lceil n/k \rceil$ VBDs are defined. For this end, we propose several k-matching backup approaches where k domains are grouped into n VBDs. These approaches intend to group a k domains into a new VBD where $k \geq 1$. For the special case of $k = 1$, each domain is also a VBD, so a new backup controller is selected for each domain/VBD. As the value of k increases, the number of required backup controllers decreases. However, the recovery time is expected to increase due to the larger sizes of the VBDs. Therefore, the proposed approach tries to strike a balance between two contradicting metrics: monetary cost (in terms of the number of required backup controllers) and recovery time (to recover from a primary controller failure).

Virtual Backup Domain: Example

Figure 22 illustrates an example of a SDN where the backup controllers are determined using VBDs. In this figure, we have an SDN network with eight domains, each has a primary controller designated by (yellow color), each manages three switches. As the architecture relies on an in-band strategy, we can observe that each primary controller is installed at a switch. Domain 1, domain 2, and domain 4, for instance, are grouped in VBD-1, which colored in blue since they are neighbouring with minimum delay time.

The algorithm computes the ideal placement of the backup controller in the VBD based on the node with minimum average response time to all nodes in the virtual domain. For $k = 3$ as an example, in the above figure, the switch S2 in domain 1 selected as the best location of the backup controller (red color) that satisfies the resilience requirements. Likewise, in the second virtual domain VBD-2, which is described with pink color, switch S1 in domain 3 selected as a candidate backup controller placement. Furthermore, for larger k values, the algorithm complexity increases significantly due to an increasing number of possible selections of k domains. To deal with the complexity issue, we developed two different categories of algorithms to determine VBDs. First, in Section 4.4.2, we design and implement a full enumeration algorithm that employs an exhaustive search for all possible domain combinations to generate the optimal grouping of domains into VBDs.

Second, due to the high complexity of the optimal algorithm (in particular for larger k and larger networks), in the three following sections, we propose three heuristic algorithms for determining VBDs based on using the node degree while minimizing the distance between selected domains. For all four algorithms, once the first phase is complete and the VBDs are determined, the specific node

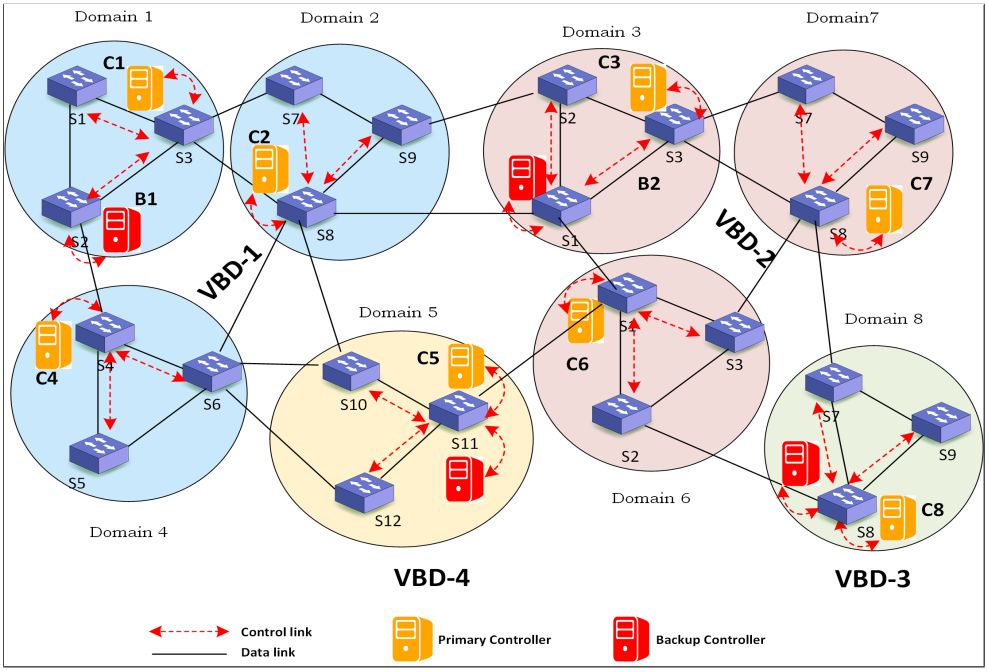

Figure 22: Backup controller placement strategy using VBDs

that will serve as the backup controller in each VBD is defined in the same way as the second phase of HDIDS (e.g. node selected provides the minimum total of the shortest paths lengths to all other nodes within the VBD).

4.4.2 Full Enumeration (FE)

Full enumeration methods are used to solve combinatorial optimization problems. Combinatorial optimization problems are problems where decision variables are binary, expressing that an object (e.g., graph, edge, node) is chosen or is not chosen. This leads to a vast number of possible solutions with the difficulty of selecting and finding the best solution [51]. This method generates all the possible combinations of k domains in the network. It measures the cost of each group and returns the best one. Since this solution approach is exhaustive, it obtains the optimal result.

In this algorithm, after determining the domains with primary controllers using HDIDS, we apply the full enumeration technique on all domains in the network where its primary controller placement will represent each domain. The method relies on measuring the cost function defined as the shortest distance between the domains (primary controllers). Generally, adjacent domains are more connected, therefore the algorithm will return all possible solutions of combinations of k domains.

4.4.3 Max Degree with Short Distance (MDSD)

In the MDSD algorithm for determining VBDs, if there is only one domain with a maximal degree, then this domain is selected. If there is more than one domain with a maximal degree, the domain with the smallest total of the shortest path lengths from its primary controller to all other domains (primary controllers) is chosen. Algorithm 2 summarized steps of our first proposed heuristic algorithm. In this Algorithm, we reference some simple procedures:

- **AllDomainsWithMaxDegree**(S, G, C) which returns a set of all the domains with the maximum degree in S.
- **DomainMinTotalShortestPaths**(K', G, C) which returns a domain with the smallest total of shortest paths from its primary controller to all other primary controllers in S.
- **AControllerMinShortestPath**(S, T, G) which returns a domain from set S that is a neighboring domains of any domain in P with the minimum shortest path from its primary controller to any primary controller representing the domains in the set of domains P.
- **Tuple**(k, T) which returns the set T as a k-tuple, padded with empty entries if $T < k$.

Algorithm 2 MDSD

Require: C, $G(V, E)$, k

1: $S \leftarrow C$, $K \leftarrow \emptyset$, $P \leftarrow \emptyset$, $T \leftarrow \emptyset$
2: **while** $S \neq \emptyset$ **do**
3: $K' \leftarrow$ AllDomainsWithMaxDegree(S, G, C)
4: **if** $|K'| = 1$ **then**
5: $K \leftarrow K \cup (K' = \{c\})$
6: **else**
7: $c \leftarrow$ DomainMinTotalShortestPaths(K', G, C)
8: $K \leftarrow K \cup \{c\}$
9: **end if**
10: $S \leftarrow S \setminus \{c\}$
11: $T = \{c\}$; $i = k - 1$
12: **while** $i > 0$ **do**
13: $\{m\} \leftarrow$ AControllerMinShortestPath(C, T, G)
14: **if** $\{m\} \neq \emptyset$ **then**
15: $T \leftarrow T \cup \{m\}$; $i = i - 1$
16: **else**
17: $i = 0$
18: **end if**
19: **end while**
20: $P \leftarrow P \cup$ Tuple(k, T);
21: **end while**
22: **return** P

4.4.4 Low Degree with Short Distance (LDSD)

Opposed to the previous algorithm MDSD, this algorithm starts with a candidate domain with the minimum degree instead of maximum degree (see line 3) and then otherwise follows the same steps of MDSD algorithm as shown in Algorithm 3.

Algorithm 3 LDSD

Require: C, $G(V, E)$, k

1: $S \leftarrow C$, $K \leftarrow \emptyset$, $P \leftarrow \emptyset$, $T \leftarrow \emptyset$
2: **while** $S \neq \emptyset$ **do**
3: $K' \leftarrow$ AllDomainsWithMinDegree(S, G, C)
4: **if** $|K'| = 1$ **then**
5: $K \leftarrow K \cup (K' = \{c\})$
6: **else**
7: $c \leftarrow$ DomainMinTotalShortestPaths(K', G, C)
8: $K \leftarrow K \cup \{c\}$
9: **end if**
10: $S \leftarrow S \setminus \{c\}$
11: $T = \{c\}$; $i = k - 1$
12: **while** $i > 0$ **do**
13: $\{m\} \leftarrow$ AControllerMinShortestPath(C, T, G)
14: **if** $\{m\} \neq \emptyset$ **then**
15: $T \leftarrow T \cup \{m\}$; $i = i - 1$
16: **else**
17: $i = 0$
18: **end if**
19: **end while**
20: $P \leftarrow P \cup$ Tuple(k, T);
21: **end while**
22: **return** P

The reason for developing degree-based algorithms is to consider if the domain degree has an effect on the response time improvement. We further develop this idea by proposing the following algorithm that combines two domains based on inter-domain adjacent degree. It starts by calculating all short paths between the primary controllers, then combines each k domain conditioned by the direct connection between any two nodes inside those domains.

4.4.5 Inter-Domain Adjacent with Short Distance (IDASD)

The idea of this algorithm is to return closest pairs of controller representing neighboring domains.

The details of this algorithm are demonstrated in Algorithm 4 where we reference the following simple procedures: a) AllShortestDistBetweenPairs(S, G) which returns a list of all the shortest distances between each of the pairs of controllers in S; b) APairWithShortestDist(S, T) which returns, using the shortest distances between pairs stored in T, a pair of controllers in S with the shortest distance; c) IfNeighbouringDomains(S', G) returns TRUE is the pair of controllers S' represent neighboring domains (the shortest path between the pair of controllers has no more than two hops); and d) RemoveAllPairsContains(S, s) which removes all pairs in S which contain the controller s. Algorithm 4 returns a list of k-tuples of controllers representing neighboring domains. Any remaining unselected domains are considered VBDs.

Algorithm 4 :IDASD

Require: C, G

1: $S \leftarrow (C_i, C_j)$, $i \neq j$, $T \leftarrow \emptyset$, $P \leftarrow \emptyset$
2: $T \leftarrow$ AllShortestDistBetweenPairs(S, G)
3: **while** $S \neq \emptyset$ **do**
4: $S' \leftarrow$ APairWithShortestDist(S, T)
5: $S \leftarrow S \setminus S'$
6: **if** IfNeighbouringDomains(S', G) == TRUE **then**
7: $(s, t) \leftarrow (S')$
8: RemoveAllPairsContains(S, s)
9: RemoveAllPairsContains(S, t)
10: $T = \{s, t\}$; $i = k - 2$
11: **while** $i > 0$ **do**
12: $\{m\} \leftarrow$ AControllerMinShortestPath(C, T, G)
13: **if** $\{m\} \neq \emptyset$ **then**
14: $T \leftarrow T \cup \{m\}$; $i = i - 1$
15: RemoveAllPairsContains(S, m)
16: **else**
17: $i = 0$
18: **end if**
19: **end while**
20: $K \leftarrow K \cup$ Tuple(k, T);
21: **end if**
22: **end while**
23: **return** K

4.5 Performance Evaluation

In this part of the experiments we review the results obtained from of the survivable backup mechanism. All experiments results were carried out on an Intel(R) Core(TM) i7-6770 CPU @3.40GHz and 16GB RAM with Windows 7 Pro (64-bit) operating system.

4.5.1 Survivable Backup Controller Placement

After determining the appropriate placements of the primary controllers within the network, the next step is to set the backup controllers' locations carefully. This process relies on 1) the best joint-domains approach using the full enumeration and the heuristic algorithms, 2) find the most appropriate location of backup controllers within the virtual backup domains. For this purpose, we contrast the average response time results for both FE and the heuristic approaches. Afterward, we show the effect of the average response time of the selected placement of the backup controller to all nodes in the VBD. Finally, we examine the quality of the solutions we gained by all algorithms.

Average Response Time of k-packing Approaches

With the varying number of k domains, in Figure 23, we observe that for $k = 2$ the FE solution earned the best average response time with 34.01 ms. In comparison, the IDASD algorithm came in the second best solution with a slightly different of 34.25 ms. On the other hand, $k = 3$, the average response time of the MDSD is the best average response time with 26.44ms, while the FE algorithm is the worst average response time with 58.28ms.

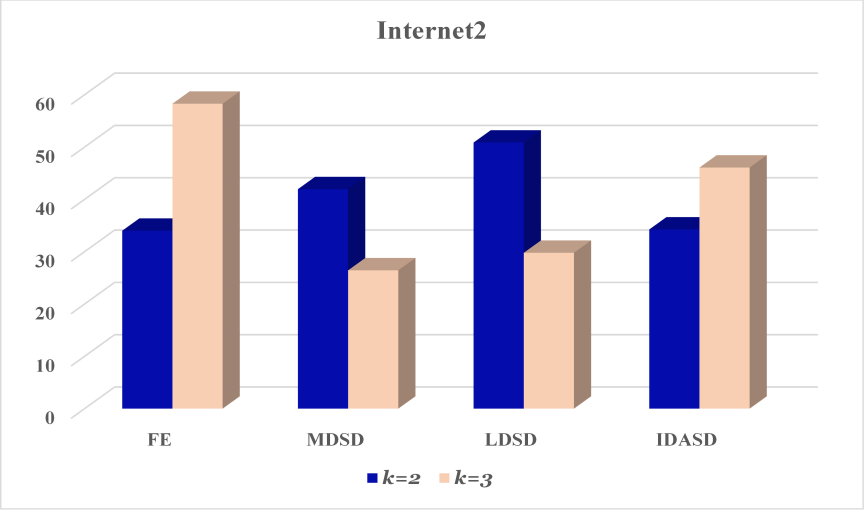

Figure 23: Average response time of k-packing grouping method for Internet2 topology

Concerning the effect of the topology structure, Figure 24 of TataNID topology shows the difference in the performance compared to the Internet2 topology. At k=2, the best average response time with 15.56ms still given by the FE algorithm. However, MDSD has performed the best performance among all the heuristic algorithms with 15.68ms. Increasing the number of joint-domains reflects the better performance of our heuristic algorithms. It can be noticed that the best average response time was achieved by MDSD algorithm with 14.09ms followed by IDASD with 14.21ms.

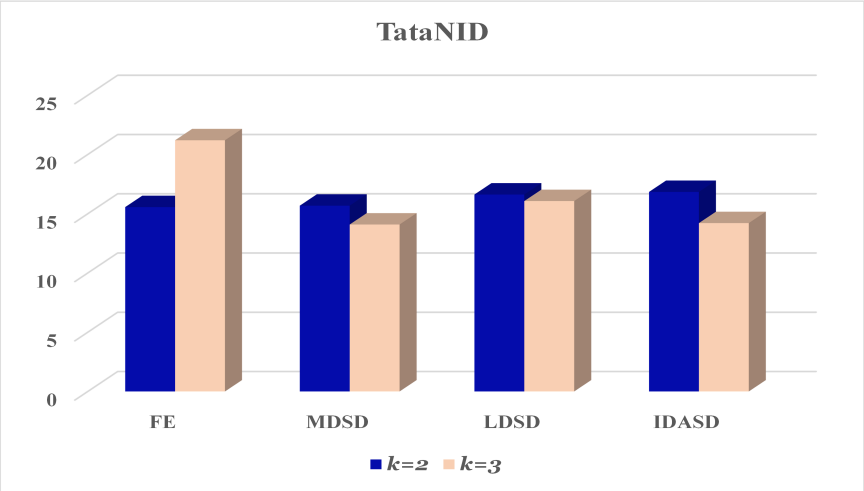

Figure 24: Average response time of k-packing grouping method for TataNID topology

Average Response Time of selected Backup Controller Placement

Here we examine the differences between the optimization and heuristic algorithms in terms of selecting the survivable controller node. In order to find this node, we perform a straightforward technique, which is to choose the node that provides the minimum total of the shortest paths lengths to all other nodes in VBD to be the survivable controller placement.

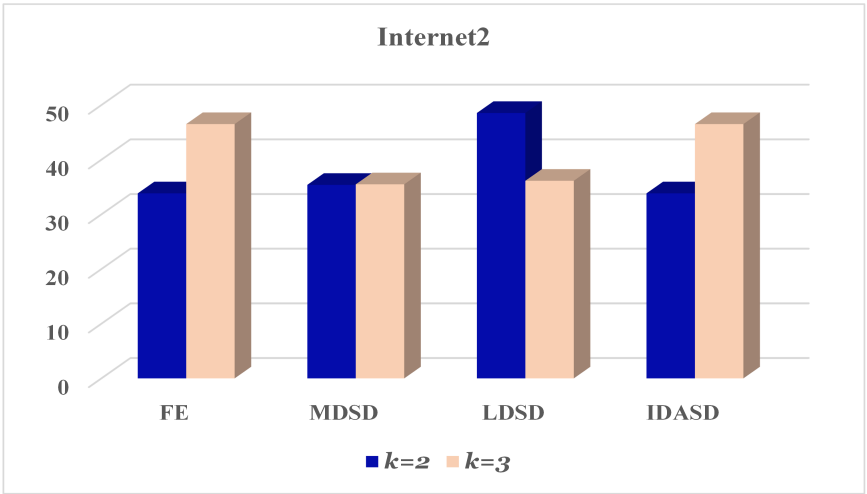

Figure 25: Average response time of backup controller placement in Internet2 topology

In terms of medium networks, Fig. 25 demonstrates Internet2 topology with k=2 and shows that our adjacent heuristic algorithm IDASD achieved 33.85 ms of average response time as the FE algorithm. With maximize the number of joint-domains k to 3, the FE and IDASD algorithms gained the worst average response time in contrary to their performance when $k = 2$. In contrast,

the MDSD algorithm experienced a lower average response time of 35.5 ms, displayed by a blue bar.

The experiment results in Fig. 26 exhibits the average response time in terms of selecting the backup node in TataNID topology. At k=2, the FE algorithm gives 12.9 ms as the best average time, while the heuristic algorithms' performance is close to each other with slight differences. This is due to the fact that our heuristic algorithms rely on the nodal degree and adjacent mechanism, and because 85% of the nodes degree in TataNID topology range from 3 to 5 links. The better average response time acquired by the heuristic algorithms was 15.45 ms by the MDSD. As can be seen in the same figure (beige bar), with an increase k to 3, our heuristic algorithm MDSD obtained the best average response time with 15.85 ms.

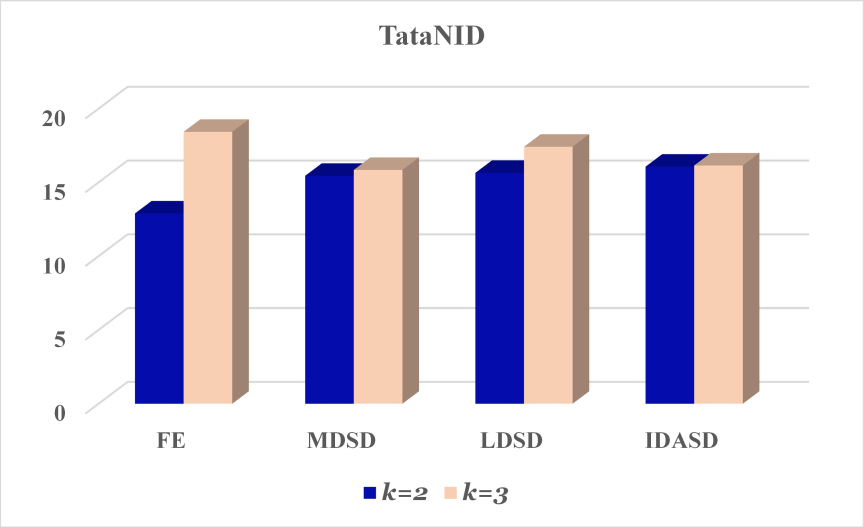

Figure 26: Average response time of backup controller placement in TataNID topology

The reasons behind this are concluded in two points: 1) as stated, the complexity of the network is related to the number of nodes and 2) increasing the number of clustered domains. In addition, the heuristic algorithms are restricted by some conditions to come up with the near FE solution quickly.

It should be noted that the computation time of FE algorithm is considerable for larger topologies and value of $k = 4$ and more. Figure 27 shows the simulation run time for the all algorithms on Internet2 topology with k values equal to 2 and 3. we can see that FE algorithm takes run time longer than the heuristic algorithms even in the seconds.

The same time consumed by all algorithms for TataNID topology is indicated in the Figure 28. It is been clear that the simulation time increased compared to the previous topology. For FE algorithm, it reaches 167.11 seconds for $k = 2$ and 184 seconds for $k = 3$, while it did not exceed 30 seconds in all heuristics algorithms.

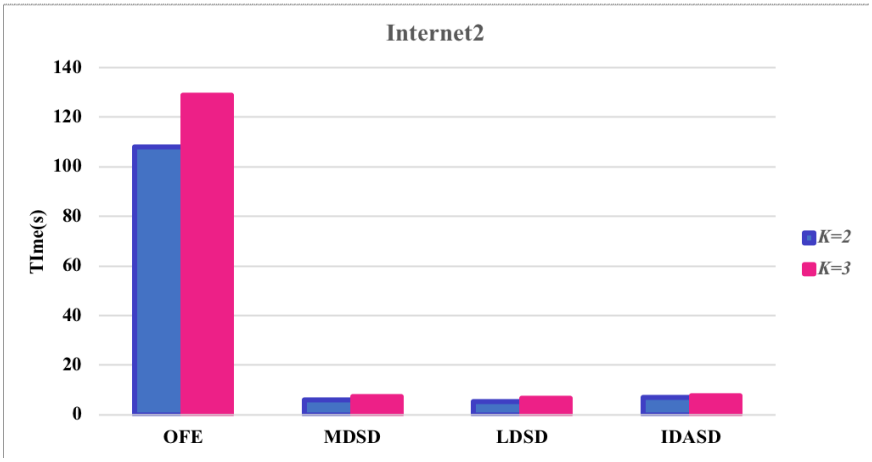

Figure 27: Simulation Run Time - Internet2 topology

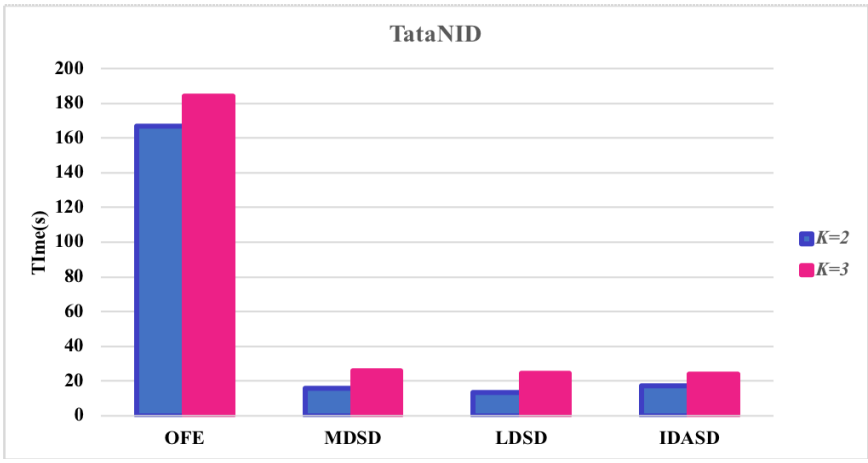

Figure 28: Simulation Run Time - TataNID topology

Quality of Solutions

As stated previously, the main goal of the proposed solutions is to combine nearby domains into a VBD. Each VBD includes up to k original domains. Finally, we seek to obtain the ideal backup controller placement that minimizes the average response time. To evaluate the performance of the solutions, we represent each domain as a node where the primary controller placement expresses this node. We consider the average delay time $f_{1,i}$ as a first cost term to measure the quality of each ith scheme. This term aims to minimize the average response time. However, this term alone is not sufficient to fully describe the quality of the resulting VBDs. E.g., a scheme may construct more VBDs from other schemes and hence decrease the average delay but at the cost of too many VBDs. Therefore, we add a second cost term $f_{2,i}$ that represents the number of generated VBDs. This term aims to balance $f_{1,i}$ by limiting the number of VBDs. For each scheme (i), both terms

report different ranges of values. For example, the number of VBDs ranges from 1 to the maximum number of controllers, whereas the delay time is measured in milliseconds. To be able to combine both these cost terms into an effective cost function, we first normalize both the cost terms into the range $[0, 1]$. Hence, the cost function for scheme i is computed as:

$$F_i = \alpha \left(\frac{f_{1,i}}{\max\limits_{allj} f_{1,j}} \right) + (1 - \alpha) \left(\frac{f_{2,i}}{\max\limits_{allj} f_{2,j}} \right) \tag{3}$$

where α is used as a weight coefficient to linearly interpolate between the two normalized cost terms. In case of α is equal to 1, the cost function is equal to $f_{1,i}$ whereas when α is equal to 0, the cost function is equal to $f_{2,i}$. With this equation, we strike a balance between two conflicting factors: performance and cost.

The effect of changing the value of the parameter α varies. A lower number of VBDs means a lower cost for the network operators (fewer backup controllers will be needed) and higher delays experienced by the users. On the other hand, a higher number of VBDs means a higher cost for the network operators (more backup controllers will be needed) and lower delays experienced by the users as a result of the tight grouping of clusters.

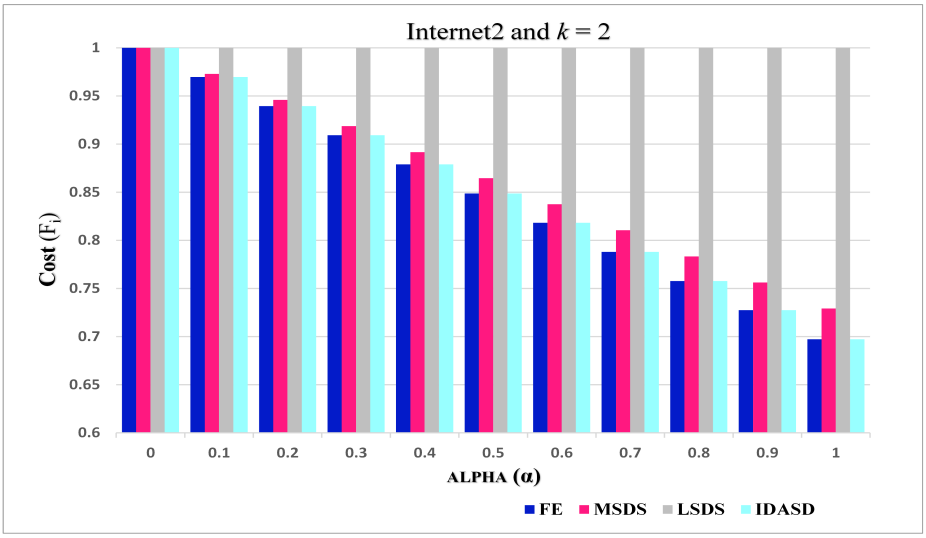

Figure 29: Quality of proposed solutions for $k = 2$ in Internet2 topology

Figure 29 illustrates the results for Internet2 topology and for k equals to 2. We note that all schemes produced the same number of virtual clusters; hence, $f_{2,i}$ has no effect on the total cost function F_i. It shows that both the FE and IDASD schemes achieved the best solution for the Internet2 network. The results for increasing the number of combined clusters from 2 to 3 are depicted in Figures 30.

When α equals to 0 (i.e., the cost function is solely evaluated using the number of generated

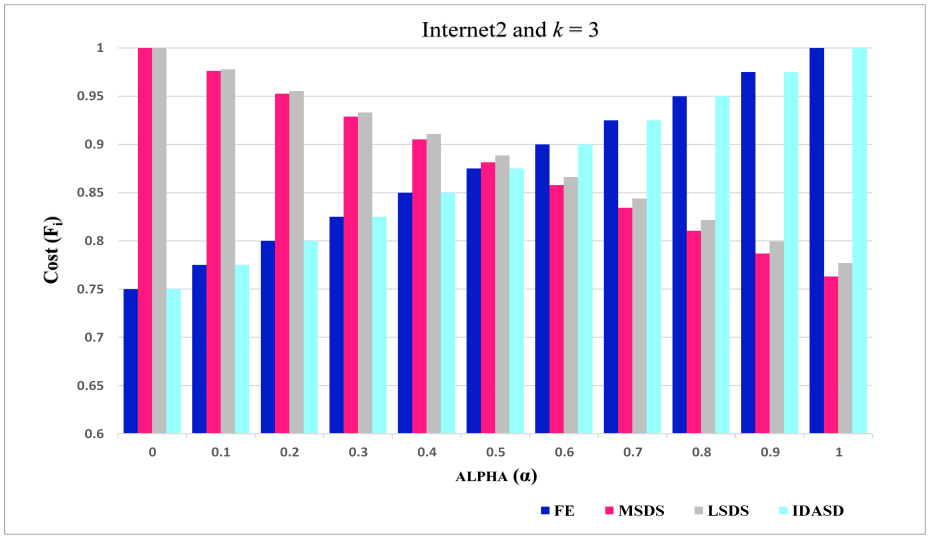

Figure 30: Quality of proposed solutions for $k = 3$ in Internet2 topology

VBDs $f_{2,i}$), both FE and IDASD schemes achieved the best solutions as both schemes resulted in a lower number of VBDs than MDSD and LDSD. However, when increasing the value of α, both MDSD and LDSD schemes managed to outperform the FE and IDASD schemes as MDSD and LDSD achieved lower delays. As the value of α increases, the effect of the delay factor $f_{1,i}$ increases. The experiments' results of TataNID topology are demonstrated in Figures 31 and Figure 32 for k = 2 and k = 3 respectively. For the case of k = 2, we note that all schemes resulted in the same number of VBDs, hence, there is no effect of $f_{2,i}$ on the total cost F_i. However, we note that the FE scheme achieved the best solution followed by MDSD.

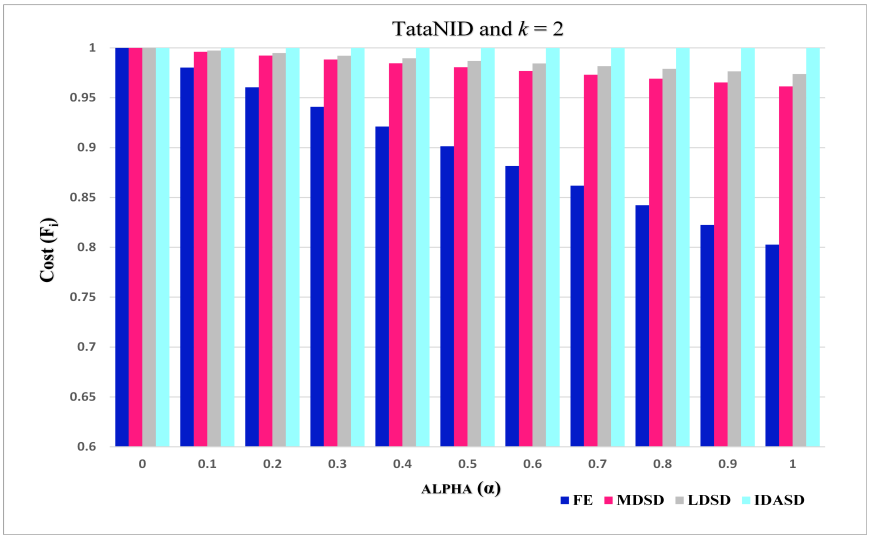

Figure 31: Quality of proposed solutions for $k = 2$ in TataNID topology

For the case of $k = 3$, the FE algorithm achieved the minimal number of VBDs, hence, a higher delay value. The LDSD scheme achieved rather balanced results in terms of both parameters. The MDSD and IDASD achieved comparable results.

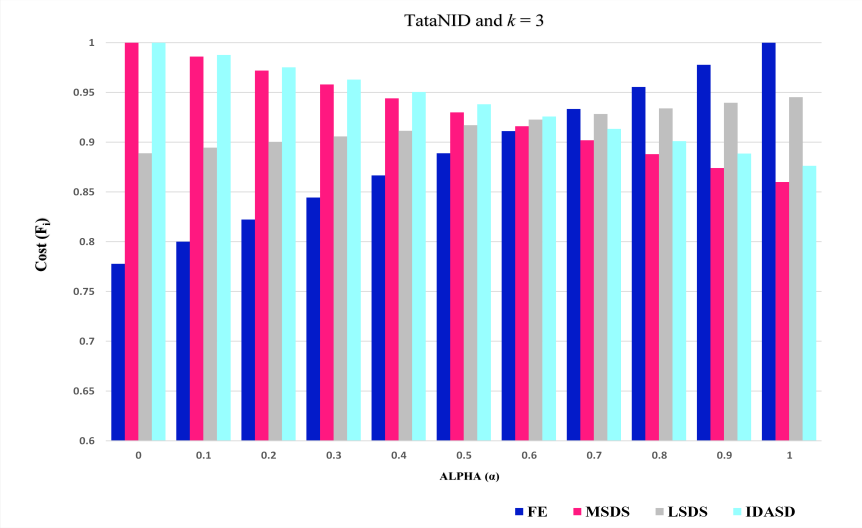

Figure 32: Quality of proposed solutions for $k = 3$ in TataNID topology

4.6 Conclusions

As networks grow today, managing networks with a single controller is hard due to the increasing number of switches that require a large number of paths through which data is routed to all components of the network. Preserving a network of a large number of switches from the failure requires an efficient mechanism to deal with the placement of primary controllers and, at the same time, make network controllers robust enough to handle sudden failures, maintain availability within a permissible time frame. To improve the network's trustworthiness and its robustness under failure circumstances and ensure the stability of the network, in this chapter, we have proposed a survivable backup controller placement approach that enhances the performance and throughput of the network. We evaluated our algorithms comprehensively using real topologies. The experimental results illustrated that our approaches provide better results, especially in large networks, in minimizing response time reduction.

Although SDN is a promising technology and has positive influences on network performance improvement, many challenges are becoming evident and need more research and effective solutions. On the one hand, the failure of this in-band SDN network has a significant impact on performance, especially when the failure is tied to the main controller. Motivated by this matter, in the next chapter, we will discuss the failure of the main controller under the in-band controlling design in the SDN networks.

Chapter 5

Dynamic Recovery Module For In-band Control Channel Failure In Software Defined Networking

This chapter proposes the implementation of our novel architecture and its core that is embodied in the ICPM module. In this chapter, we formulate the in-band control channel failure problem, we propose dynamic failure recovery methods and measure the reliability of the control channel in in-band SDN networks.

5.1 Introduction

SDN is an innovative paradigm that provides more flexibility and adaptability of data communication by separating the control plane from the data plane. In SDN, controller placement involves assigning a controller to a set of switches in a given portion of the data plane. OpenFlow is a standard protocol that defines how to create and manage the connections between the controller and switches in SDN [52].

Despite the advantages of the in-band control structure, the usage of in-band control management suffers from a weakness regarding failure occurrences. For instance, when a failure occurs in a control component (node/link), the subsequent nodes will be affected by such failure. Consequently, control packets may not be processed correctly in the control-lost switches, hence leading to performance degradation. Furthermore, the master gate switch, where the controller is placed, may fail as any part of the network due to the hardware failure, attack, etc. A single node failure can adversely affect other neighbouring nodes that depend on the defective node and could fail. This phenomena called

a *dependency network* [53]. See an example shown in Figure 33 where the routes to the controller of the nodes 2, 3, and 5 are dependent on node 4; if node 4 fails, all the aforementioned nodes will be affected.

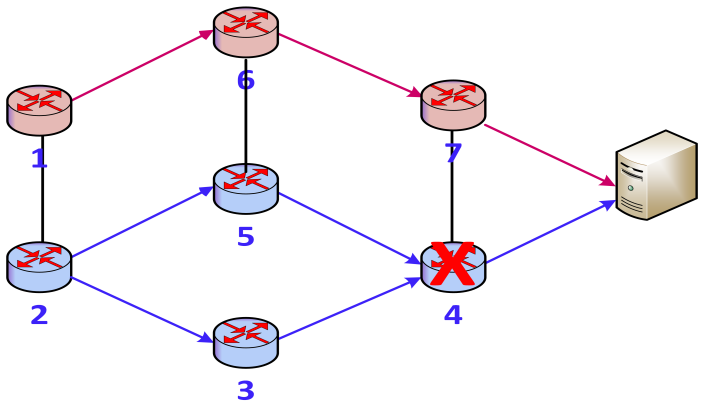

Figure 33: Failure in dependency networks

In in-band SDN networks, this matter is crucial, so dealing with control channel recovery after failure is one of the in-band network challenges that need to be investigated. Motivated by these factors, the objective of the chapter is to deal with in-band failure occurrence in SDN effectively. In particular, our goal is to provide more reliability and controllability of in-band SDN fashion under the failure scenario, and therefore we propose an In-band Control Protection Module (ICPM). The ICPM provides a protection approach of the in-band control channel that addresses the failure of any of the in-band control component in SDN networks and protect as much as possible of the control traffic of the affected switches. The ICPM approach does not apply to out-of-band control architectures since when a switch fails in an out-of-band SDN, the controller will discover this failure and immediately react to the failure by redirecting the control traffic to the backup link either by a proactive or reactive strategy [54].

The contributions of this chapter are threefold. **First**, we formulate the in-band control channel failure problem where we propose dynamic failure recovery methods and measure the reliability of the control channel in in-band SDN networks. **Second**, we define two notions: neighborhood and backup, based on which we give the formulation of the problem and design a module integrated into the switches to solve the problem. **Third**, we conduct experimental simulations to show our approach is efficient compared to the benchmark of other works [55], and evaluate the performance in networks of different scales and degrees of connectivity.

5.2 Related Work

In the out-of-band mode of network control, when a controller fails in the physical network, all the switches managed by the failed controller are disconnected from the control plane. To deal with this situation, the OpenFlow standard defines a switch as having a backup controller called a slave controller such that the switches can migrate to the backup controller easily. However, dealing with the issue of failure in in-band networks is different because even a (link/switch) failure may impact the control system of the network and result in more disruption compared to the out-of-band scenario [56].

There exists some research work on how to make in-band control for SDNs practical, including work on failure detection and recovery for paths between switches and controllers. Sharma *et al.* [54] studied the problem of link failure that occurs in the data plane and how it affects the control communication between the switches and the controller. For this, a restoration mechanism was presented to make the controller react to link failure by removing all affected forwarding rules and installing new rules after computing the backup paths. In addition, Sharma *et al.* [54] proposed a protection mechanism where all backup paths are pre-established before any link failure such that traffic will be immediately redirected to the backup paths when the failure occurs. Their results show that only the full protection recovery scheme meets the requirements of 50ms [57]. The authors of [54] did not consider situations where the controller or master gate switch fails.

A flow tagging mechanism introduced by Thorat *et al.* [58] is a proactive recovery method to recover from single link failure. In the event of a link failure, all victim packets will be tagged with a VLAN label and redirected to the other end of the failed link. There are several techniques for achieving immediate recovery from failure. One of the common techniques employed by today's networks is protection recovery mechanism. Generally, the protection methods are classified into two main categories: link protection or path protection.

Further, path protection is classified into two types, which are 1:1 and 1+1 [59]. 1+1 path protection requires provisioning two disjoint paths where the data traffic generated by the source node sends over both paths. In case of failure, at least one of the paths remains alive, hence the destination node will be able to receive the data without interruption. While the path protection scheme is simple, easy, and increases the overall throughput of the transmission, it incurs high communication overhead due to doubling the transition of data over two disjoint paths. Conversely, the 1:1 path protection technique employs only the primary path to transmit traffic. However, in the event of a link or switch failure, the 1:1 path protection technique redirects the traffic from the failed primary path to an alternate backup path [60].

Most of the recent works exploring failure management in SDNs have relied on constructing a

substitute path for every possible disrupted flow [61, 62, 63].

In Sgambelluri *et al.*'s work [62], there exist flow rules on the backup path for every single possible disrupted flow on the primary path. For every new incoming flow to each link along path, the controller installs new flow rules to setup the backup path. However, this approach is impractical for a network having thousands of flows per link since the approach may impose more overhead on the switch to store additional flow rules for a backup path for every flow on the link. Furthermore, the controller intervention during the rule's installation increases the recovery time due to the propagation delay of the failure notification message. Kim *et al.* [64] proposed CORONET as a fault-tolerant system that recovers from multiple link failures. The authors considered the failure event in data plane using link layer data packets (LLDP) protocol for detecting failure [64]. However, LLDP may not be efficient due to the constant monitoring requirements, which may create extra LLDP packet processing at the controller and, hence, increase the traffic in the network. Asadujjaman *et al.* [52] introduce an in-band control channel approach that deals with recovering multiple failures simultaneously.

Local rerouting and constrained reverse forwarding strategies were proposed by Hu *et al.* [55] in their protection mechanism for control traffic. In the former strategy, the control traffic is redirected to an upstream neighbor switch that is connected to the controller in its primary path, while in latter, the control traffic is forwarded to the downstream switch connected to the controller. Although control traffic may need to be forwarded back multiple hops to reach the controller, this method limits the number of hops. Therefore, it cannot guarantee that all switches will be protected.

Moreover, only a single link failure is studied in most of the related work. Ochoa-Aday *et al.* [65] presented a fault-recovery mechanism for control path in failure event. In this mechanism, the local switches react to recover the control paths quickly, and the global controller knowledge helps to optimize the recovery paths. The scheme proposed by Park *et al.* [66] aims to react to the in-band OpenFlow networks changes by adopting a detouring strategy. To this end, it selects the low traffic paths for the control messages to guarantee fast network recovery. Chan *et al.* [67] proposed a fast recovery scheme for in-band controlled multi-controller networks based on a cycle monitoring approach. With the collaboration of multiple controllers, the failure type and failure location can be determined within a short time. Goltsmann *et al.* [68] proposed a calculation scheme of the recovery path in the switches. They try to minimize the flow table size through storing only the switches-to-neighbor routes. The switches should have the ability to quickly discover the network topology and install flow entry into other switches.

In terms of dealing with failure occurrence in the data plane, some research have been conducted to deal with this issue. Capone *et al.* [14] further develop a protection scheme of networks based on backup paths computed in advance. This approach was inspired by employing the MPLS routing

protocol that guarantees minimum recovery time and zero packet-loss after failure detection. Similarly, Cascone *et al.* [69] proposed an MPLS-based method to treat the failure of the SDN data plane by distinguishing different forwarding behaviors to react upon failure without controller involvement.

The major difference of our approach compared to most of the previous works is that our proposed strategy handles failure events that may occur in the master gate switch (i.e. the switch where the controller is placed) and through which all control messages are exchanged. Also, our strategy deals with any node's failure located along the path of a control channel. On the contrary, while some methods proposed by previous works were efficient in dealing with link failure situations, they were not efficient in dealing with node failure. However, our approach module can't handle the links failure in large-scale networks due to the high volume of exchanged messages between neighbors until a broken link is detected and hence prepare a recovery path.

In addition, our algorithm will seek to minimize the amount of lost control messages during a node failure. To this end, the proposed algorithm will convert the traffic along its available path while avoiding local loops in the network until updating all routing information for all nodes along the affected path. This mechanism aims to avoid the loss of control packets, especially for real-time traffic, and improve the qualify of service (QoS). Our goal is then to deal with different failure scenarios, whereas much control traffic as possible can benefit from the proposed protection approach. In the following section, we formulate the in-band protection control problem and then introduce an approach to solve it.

5.3 Problem Formulation

In this section, we formulate the in-band control network protection problem, which focuses on defining the control networks and maximizes the number of switches, whose control packets can be protected by the proposed protection approach.

5.3.1 Network Model

Our focus in this sction considers in-band control in SDN with a single controller, where each switch has only one routing path to reach the controller. We represent the network as a graph $G(V, E)$ where V are graph nodes that denote the switches set in the network, and E denotes the links set, the connections between pairs of switches in V where the weight of each link is the propagation delay of the link. In SDN, the switches and controllers are the forwarding elements, therefore, we make the distinction in our work of assuming that the controller locations are at switch locations in the network. For each switch $i \in V$, let $C(i)$ be the controller that manages switch i, let $NG(i)$ denote set of neighboring switches of the switch i in G, and let $d = |NG(i)|$ be the degree of the switch

v_i. We consider two subsets of switches in the neighborhood of switch i, $1NG(i)$ and $2NG(i)$. $1NG(i) = NG(i) = (v_1, v_2, \ldots, v_d)$ where $v_j \in NG(i)$ is a switch directly connected to v_i, and $2NG(i) = (v_1, v_2, \ldots, v_t)$ where $v_j \in 2NG(i)$ is a switch at a two-hop distance from v_i (excluding v_i itself) and $t = |2NG(i)|$ is the total number of switches at a two-hop distance from v_i. In light of our previous work on primary controller placement [70], we consider the placement of the controller at the switch with a high connection degree, all control traffic between a controller and the switches must be built over the same data channel in the network.

5.3.2 Terminologies

In this section, we set several terminologies that we will use later in the development phase

- **Master Gate Switch (MGS)**

  A switch $i \in V$ is termed the *master gate switch* if the controller $C(i)$ placed at the switch i location.

- **Backup Gate Switch (BGS)**

  The control path of every switch $i \in V$ is the shortest path from i to the controller $C(i)$. Therefore, switch i in the network denoted as an *backup gate switch* if i is linked to the master gate switch with shortest path.

- **Failed Switch (FS)**

  Since we are dealing with the protection of the in-band control channels against the failure in SDN, to be consistent in the following section, the switch that is suffering the failure will be called *failed switch*.

- **Detector Switch (DS)**

  In case of switch i failure, a switch $j \in V$ is called a *detector switch* if it meets the following conditions:

  1. It is directly connected to the failed switch i, and
  2. Its control path passes through the failed switch i.

- **Protector Switch (PS)**

  In case of switch i failure, a switch $j \in V$ called a *protector switch* if the restore path from the detector switch for i passes through j.

5.3.3 In-Band Control Network Failure

In SDN networks, the failure of the control plane may result in a high risk to the network, especially if there are no efficient prepared steps to overcome this matter. Particularly, in an in-band controlling

scheme, the failure impacts different components such as the master gate switch where the controller placed which is the most critical part of the in-band controlling scheme or any data plane elements (node/link) that exist on control paths. In this section, we present different failure scenarios in in-band control logic in SDN.

5.3.4 Master Gate Switch Failure

In SDN networks, the control and data planes are both involved in the in-band control management. In particular, the master gate switch is the main component of in-band control responsible for passing all control traffic to the controller. While most research works suggested different ways to address the data plane failure in the in-band SDN controlling scheme, yet the problem of dealing with the failure of the most important control component (the master gate switch) still needs to be investigated. To the best of our knowledge, we are the first to consider the failure of the master gate switch. The master gate switch is a critical point because any flaw in this component dramatically influences the entire network performance level.

As stated earlier, we want the control traffic to be re-forwarded to the controller as fast as possible while preserving the maximum amount of control traffic. Therefore, we adopt building network control channels under the management of two gate switches. To this end, we use a backup gate switch beside the master gate switch to support recovering the network from the master gate switch failure.

The controller and switches can maintain proper connectivity through using OpenFlow protocol echo request (OFPT_ECHO_REQUEST) and reply (OFPT_ECHO_REPLY) messages. This message and its reply ensure continuity of communication between the controller and switches [10]. In the design of our approach, when the controller no longer receives OFPT_HELLO messages (alive messages) from the master gate switch, this means a failure has occurred. The controller immediately will activate and migrate the control traffic to the backup gate switch, update the topology, and change all control paths as needed.

5.3.5 Control Path Switch Failure

Any switch can represent more risk when a high percentage of control traffic passes through it. In other words, if there are switches that exchange the control messages with the controller over a particular switch, the failure of this component will threaten a large portion of the network. As mentioned before, in our approach, each switch will have two control paths to the controller via both the master gate and backup gate switches. In the case that any switch along a control path fails, the former switch of the failed switch along the control path will detect the failure and activate the protection mode. To make our approach more reliable, we have restricted the backup path of

each switch such that the backup path does not pass through the master gate switch and instead, we use the shortest path to the backup gate switch. To determine these shortest paths in the network, we use Dijkstra's algorithm [71] for computing these paths for every switch in the network to the controller provided that these paths do not include passing through the master gate switch.

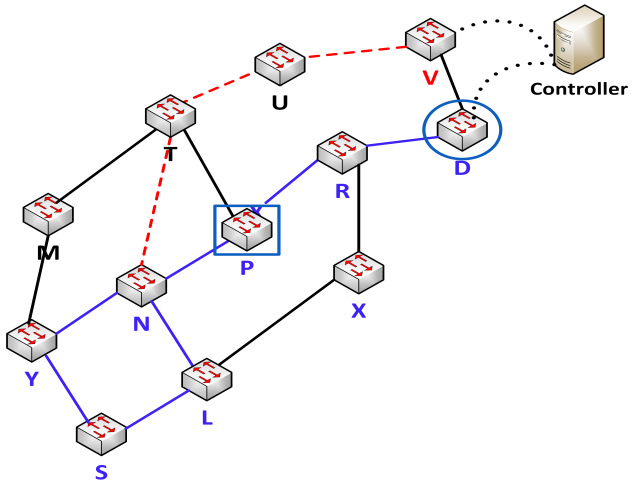

Figure 34: In-band failure scenarios in SDN networks

In Figure 34, we highlight the potential failure scenarios. We observe two switches are defined, the first switch indicated by a surrounding circle indicates the master gate switch failure, and the second switch by a surrounding square shows the control switch failure. In the first situation, as mentioned before, the controller will be alarmed by this failure and activate the backup gate switch and update the control paths. In the second scenario (again, see Figure 34), when switch P crashes, this will disable all dependent nodes (S, L, Y, and N). Hence, switch N will take action of protecting the control traffic of the affected switches by turning on its protection mode.

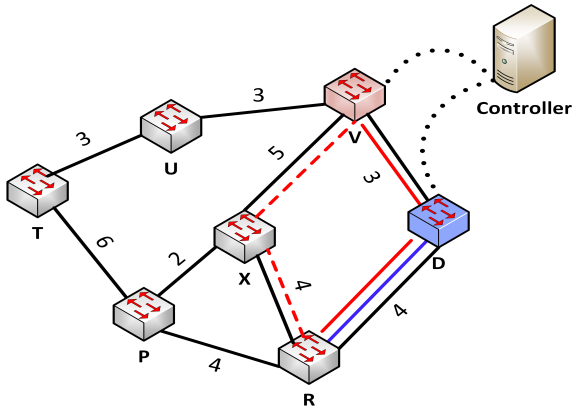

Figure 35: Backup control path illustration

For instance, in Figure 35, the controller is connected to the master gate switch D and the backup gate switch V. The primary path toward the controller of switch R as an example is colored in blue while the backup path indicated by the red solid line path which passes the master gate switch D with a cost of 7, but as aforementioned, the backup path should not pass through the master gate switch. Hence, it neglects this path and shifts to the path R-X-V with a cost of 9 as shown by a dotted red path.

5.3.6 Critical Switch Problem

Ideally, from a network management point of view, each switch should have two disjoint control paths such that the backup path might provide fast recovery of the failure of any control component along the primary control channel. However, a critical problem may still arise during the recovery process. Define the importance of a switch such that the more primary and backup paths that pass through a switch, the more important it is. Consider that the switches in the network are ranked in terms of their importance. Then, the higher the switch rank, the more critical it is [72]. The most critical switches may be located at the intersections of a large number of primary and backup paths, and therefore their failure may result in the breakdown of the entire recovery process.

5.4 In-band Control Traffic Protection Scheme

To meet the reliability requirement in in-band control traffic against unexpected control component failure in SDN, in this section, we introduce a control protection module that utilizes a distributed control traffic component for failure detection and restoration method.

5.4.1 ICPM Module Overview and Design

Module Overview

We introduce a dynamic distributed technique to protect and restore the in-band control channel in SDN. Our goal is to achieve a reliable controller-switches control channel in the presence of any single control component failure. More specifically, we propose an In-band Control Protection Module (ICPM) that accomplishes a set of tasks that depend on each other (e.g., failure detecting, preparing an appropriate recovery path, and restoring the control channel via an alternative path). Hence, the switches can handle the control channel by their own means.

Module Design

The key objective of this module is to preserve the control channel as protected as much as possible under failure circumstances while also protecting as much as possible the fail-affected control traffic. Therefore, to ensure reliable control channel recovery, we implement different functions inside the

module, which is described in the next part. Figure 36 illustrates the design of ICPM functions with the information exchanged between all units.

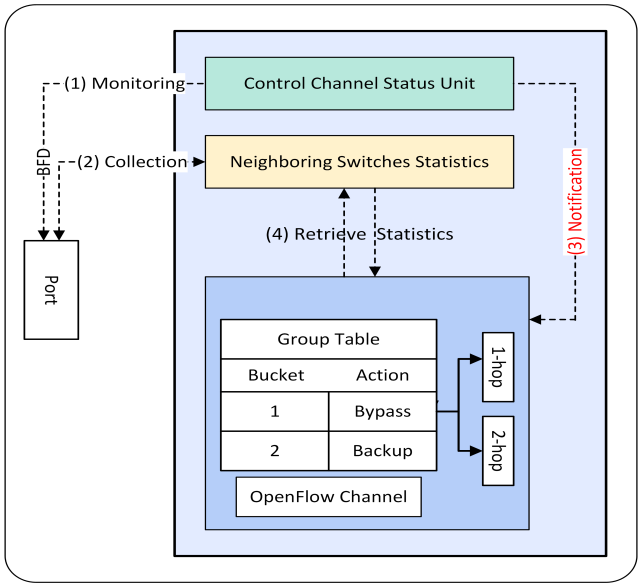

Figure 36: In-band Control Protection Module (ICPM)

- Control Channel Status Monitoring (CCSM) Unit

  This unit is in charge of tracking the port status and detecting the failure of the next control switch. The BFD protocol [73] was adopted in this unit to detect the failure between two switches. Each switch transmits bidirectional forwarding detection (BFD) packets periodically to its subsequent switch along the control path. If the switch misses M consecutive BFD packets, it deems that the other switch experienced a failure.

- Neighboring Switches Statistics (NSS) Unit

  Restoring and protecting the control channel depends on how fast and responsive the module in calculating and determining the best recovery path. Therefore, in every switch's ICPM, this unit exchanges the neighboring switches' information (e.g., location, distance, and connection).

- Control Recovery (CR) Unit

  To recover the control path after failure, the CR unit of each switch's ICPM determines the new control path to the controller based on the statistics information collected by the NSS unit. In OpenFlow protocol, a fast-failover using *Group Entry* has been introduced in order to switch the traffic to an alternative path without involving the controller [11]. Therefore, we use the Group Table concept to protect the control channel by pointing the control traffic to an alternative path. Each flow entry in the flow table guides the packets to one of the group

entries of group table which contains a number of action buckets. Precisely, we design the group entries to have two main action buckets I) Hop-Bypass mode, II) Backup Forwarding mode. In the bypass mode, the control traffic bypasses the failed control switch using 1-hop or 2-hop neighboring switches, while in backup forwarding mode, the CR unit will change the traffic to the pre-calculated backup path. Eventually, the RC unit compares the resulting paths of both buckets and ultimately decides which recovery path to choose. More details on these techniques are presented in the next section.

5.4.2 Implementation

Hop-Bypass Mode (HBM)

In the hop-bypass scheme, a switch makes a local decision to select the proper neighbor hop for forwarding control traffic. To this end, a switch must know all its 1-hop and 2-hop neighbors to select the neighbor whose distance closest to the switch located after the failed switch along the primary path. Although bypass protection is not limited to any routing protocol, the specifics of the mechanism depends on the characteristics of the underlying on-demand routing protocol.

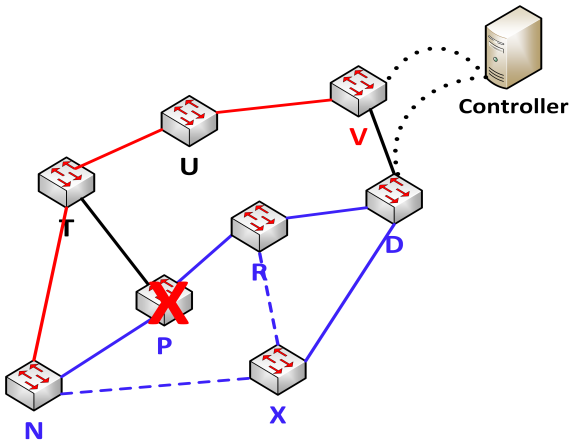

Figure 37: Hop-Bypass forwarding scheme

Figure 37, shown an illustrative example of hop-bypass protection mode. We consider the control path "N-P-R-D" from distention switch N to master gate switch D as indicated in blue color. Upon the failure of switch P, switch N as a detector detects the failure and first triggers a local query to its neighbors, repairs the failed path by using a 1-hop neighbor bypass protection mode. The neighbors reply if they have active links to switch R which is the switch located after the failed switch, if not, then move out to the 2-hop neighbor bypass mode. As shown, switch X informs switch N with its connectivity with switch R. Switch N patches the path accordingly and the control traffic is first forwarded to switch X and then to switch R to reach the master gate switch D.

Backup Forwarding Mode (BFM)

The activation of this mode relies on either the failure of the master gate switch or a switch failure along the control path where the alternate control path of the failed switch needs more than a 2-hop neighboring switch to forward the control message. Upon switch failure, the CR unit in ICPM prepares all backup paths for every switch to the backup gate switch. Finally, the shorter path will be selected, the detector switch converts the flow to the backup path as shown in Figure 37 with red color.

5.5 Performance Evaluation

5.5.1 Experimental Environment

We created our simulation platform using C++ and all the experiments described in this section were carried out on an Intel(R) Core(TM) i7-6770 CPU @3.40GHz and 16GB RAM with Windows 7 Pro (64-bit) operation system. We tested our approach on three different topologies conducted from SNDlib library [74], with their basic profiles listed in Table 5.

Network topology	Number of nodes	Number of links
Norway	27	51
Pioro40	40	89
Germany50	50	88

Table 5: Main characteristics of experimental topologies

We initialize our network by setting the required parameters that are described in Table 6. We set the delay time based on the distance calculated using nodes' latitudes and longitudes given in the topology file divided by the speed of light in optical fiber. The switch with the highest degree and with the minimum total shortest distances to other given switches in topology is configured to be the master gate switch, while the closest connected switch to the master gate switch is selected as the backup gate switch.

5.5.2 Evaluation Scenario

We generate *packet_in* messages with a constant rate of 100 packets/ms at each switch and log the time when the master gate switch receives these messages. The metric that had been used during the simulations is the number of packets that were dropped due to the failed switch divided by the total number of packets that would be sent ideally on the control channel without loss; we call the metric the Control Packet Drop Rate (CPDR). For experiments, the time from failure detection

Network topology	Value
Simulation Time	60 sec
Packet_in Transmit Rate/ms	100 packets
Experiments per switch	30 times
Failure Interval Time	Random
BFD messages	3 messages

Table 6: Simulation Parameters

(e.g., no packets pass the failed switch) until resuming send *packet_in* messages is indicated as the Control Channel Recovery Time.

We follow the same steps as described here. At first, we setup the control channel for every switch in the network, we then establish control channels from all switches by sending *packet_in* messages to the master gate switch where the controller is located. After these initial setup finished, we disconnect a specific switch at random interval time. Finally, we continue to transmit the *packet_in* messages from all switches which use the broken switch to pass its traffic until the desired switch's control channel are restored. Separately, we conduct the simulation of the same experiment with 30 runs on every switch (where the selected switch will be the one to fail). The interval time of failure selected randomly and reported the average values of lost packets number and the number of packets delivered on the master gate switch. As such, we repeat each set of experiments once for each of the switches in each topology.

5.5.3 Control Packet Receive Rate

First, we study the effect of switch failure has on the percentage of control traffic that arrives at the master gate switch. As we mention above, the total number of packets that would be sent ideally by every switch, without any switch failure, is computed according to the simulation time and the distance from each switch to the master gate switch. Figure 38 for the Norway topology indicates the percentage of the total packets ideally received by master gate switch (Rcv-ideal-stu) compared with the number of packets received in failure situation without applying our approach (Rcv-failure-stu). The ideal receive rate is less than 100% for most nodes since some packets are still in transit when the simulation ends. Note that the master gate switch is switch N16 while the switch N24 is selected as the backup gate switch. In this figure, we can notice each node's failure effect on the whole network. For instance, without applying our approach, node N9 failure can affect 20% of the network packets while the failure of node N16 which is the master gate switch can affect almost 100% of the network packets.

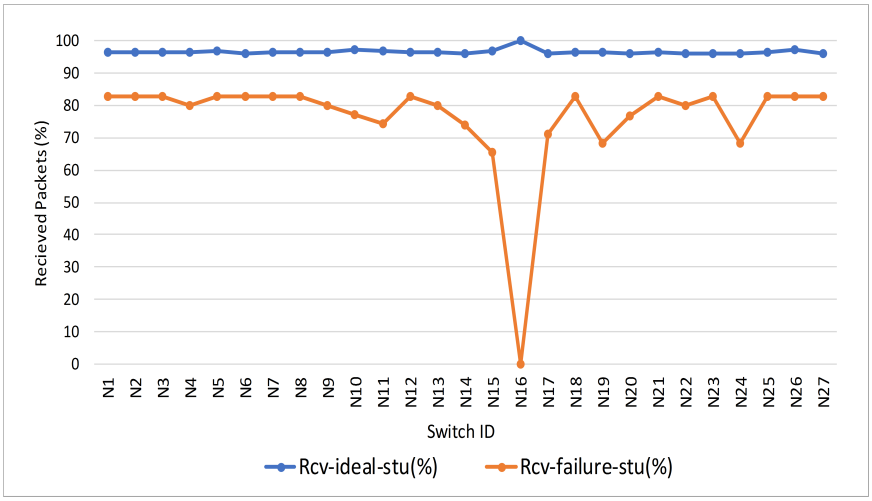

Figure 38: Packets received (Ideal vs Failure):Norway Topology

As for the Pioro40 topology, the percentage of the total packets ideally received by master gate switch (Rcv-ideal-stu) compared with the number of packets received in failure situation without applying our approach (Rcv-failure-stu) is given in the Figure 39. In this topology, node 37 selected as a master gate switch, we can see that all control packets are dropped 100%, and with the same behavior, some nodes dropped packets with different percentages, due to the fact that some packets are still in transit at the failure moment.

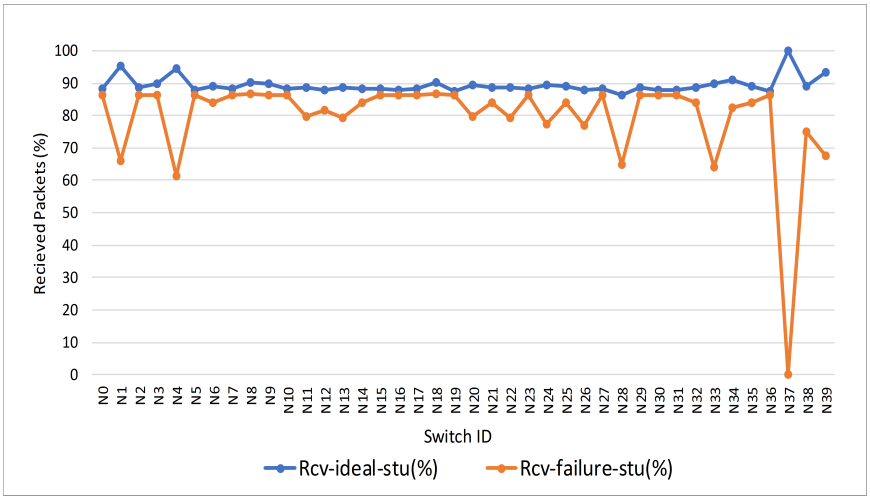

Figure 39: Packets received (Ideal vs Failure): Pioro40 Topology

5.5.4 Control Packet Drop Rate

Now, we study the improvement that our ICPM approach has on control traffic when recovering from a switch failure.

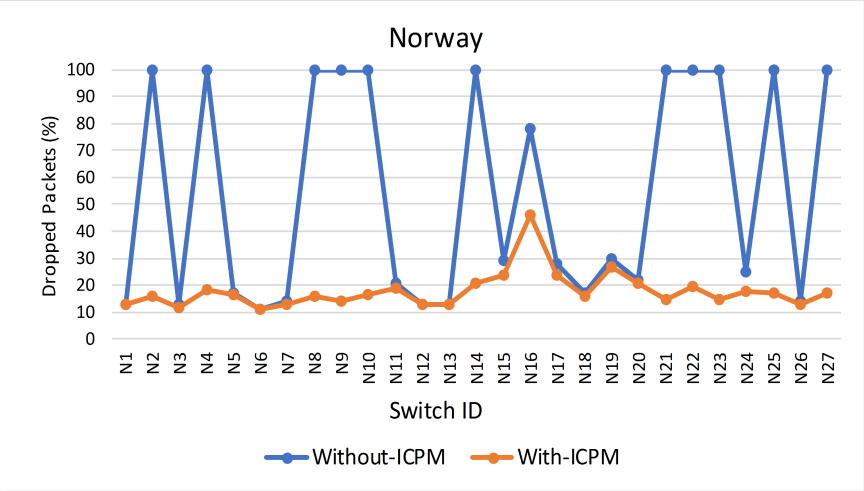

Figure 40: Control packets drop rate: Norway Topology

Figure 40 clearly depicts that the number of control packets dropped in some switches for the Norway topology. We can observe there are 11 switches in the Norway topology dropped packets at full rate of 100%. The reason behind this is as we mentioned earlier in Section 5.3.6 regarding the Critical Switch Problem. In order to demonstrate the importance of our ICPM approach and to provide a solution to this problem, we enhance our approach by proposing a backup path that is entirely separate from the primary path, meaning the backup path does not intersect the primary path for any switch. A significant improvement compared with the previous method can be seen in Figure 40 as highlighted with the orange curve. Some switches show fewer dropped packets rate than others in the control path protection scheme. This is due to the avoidance of extra overhead of the switches located far from the master gate switch or due to its less-dependent switches.

Similarly, in Figure 41, for the 25 switches in the Pioro40 topology that has previously dropped their traffic at 100%, by applying our ICPM approach, a noticeable improvement can be seen. The rate of dropped packets at some switches such as (N1, N28, and N33) appears higher than the rest, this due to a larger number of switches that rely on using these switches to send their traffic to the controller.

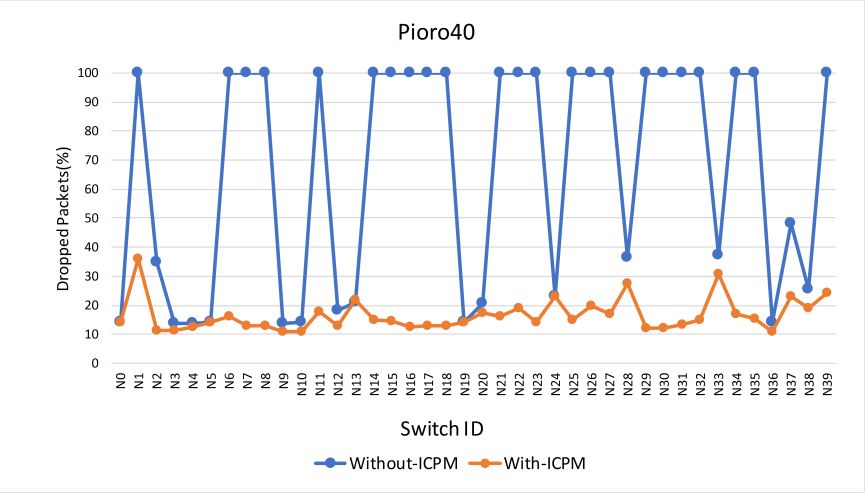

Figure 41: Control packets drop rate: Pioro40 Topology

5.5.5 Failure Recovery Time

When the switch is located on the recovery path, the control channel's restoration process can fail at that point. Visually, as shown in Figure 42, the x-axis shows switch IDs, and the y-axis shows the required time in milliseconds to setup the recovery path for protecting all the control traffic traveling by every switch. We measured the failure recovery time for all switches with randomly varying failure times. It can be seen that the recovery time of some switches is indicated by red circles which means the recovery process failed at those switches (they are not 0ms) due to the critical switch problem. In contrast, by applying the enhanced version of our methodology, we can notice the recovery time can now be seen for these critical switches. For instance, the recovery time of switch N25 changed from no-recovery to 3ms which is determined as the lowest recovery time achieved within the set of critical switches, while the highest recovery time of switch N2 changed from no-recovery to 9.8ms. From another angle, as an illustrative instance, the recovery time of switch N15 in Figure 42 was 2.1 ms in the regular recovery algorithm while it increased to 4.3 ms after applying the enhanced version of the algorithm. Again, this due to the different backup paths selection. Figure 43 shows Pioro40 topology with the critical switch matter. Switch N31 indicated as the lowest achieved recovery time of 2.5ms while switch N23 is the highest recovery time with 9.03ms within the critical switches set. Furthermore, the recovery time of switch N4 is 0ms, and the reason is that the switch N4 is the backup gate switch of the recovery process in the network.

We also consider a comparison of our approach with the reverse forwarding protection approach, which we refer to as CNP, introduced by Hu *et al.* [55]. Figure 44 presents a failure recovery time comparison. It can see that our ICPM approach achieved the lowest recovery time, while the recovery time achieved by reverse forwarding protection (CNP) is higher than our algorithm except for a few

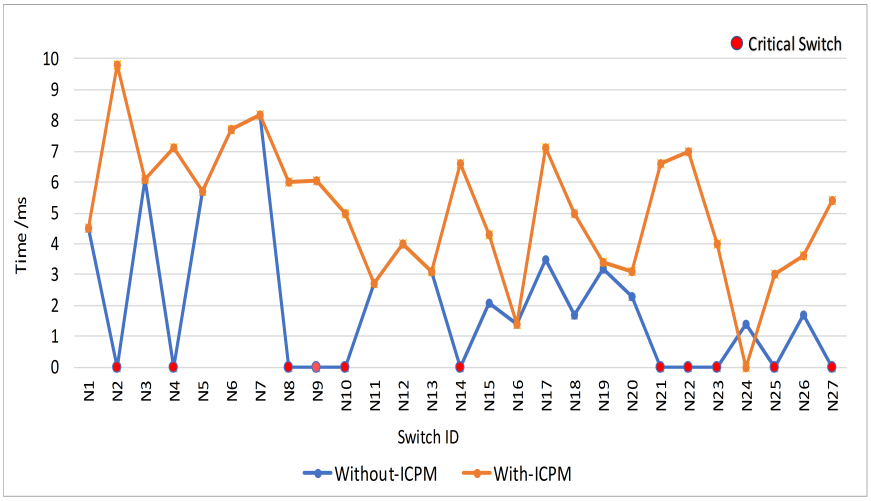

Figure 42: Failure recovery time within critical switches: Norway topology

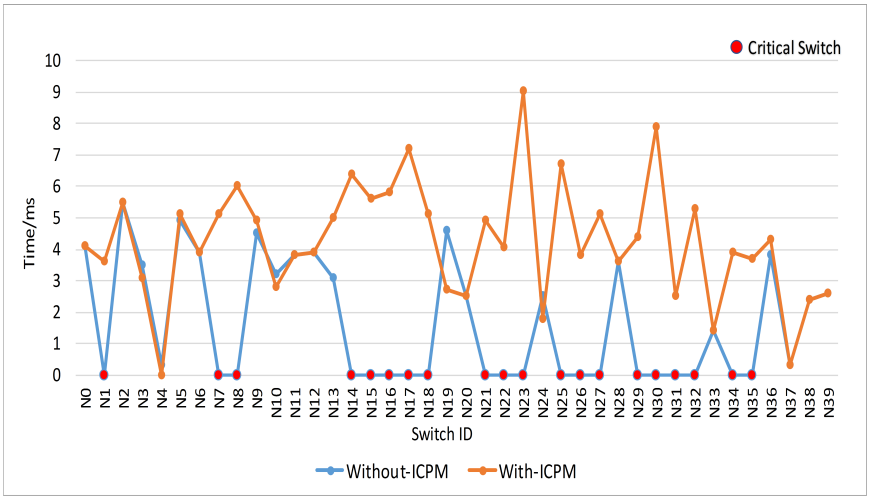

Figure 43: Failure recovery time within critical switches: Pioro40 Topology

switches that match the same recovery time. As shown, the recovery time of some switches was affected by using a non-intersecting backup path strategy. Furthermore, some other switches adopt the hop-bypass approach as a shorter path to reach the master gate switch.

In a different direction, switches (N1, N3, N5, N6, N7, N9, N11, N12, N13, and N16) maintain the same recovery time. This is due to the fact that there are no hop-bypass paths or due to the backup path that does not intersect with any switch located on the primary path. However, our ICPM approach is still outperforming the reverse forwarding protection scheme. Significantly, we can see the recovery time of switch 24 is 0ms, this due the switch N24 is the backup gate switch in Norway topology as mentioned early. In the failure situation, switch 24 immediately adopts the control traffic and directs it to the controller.

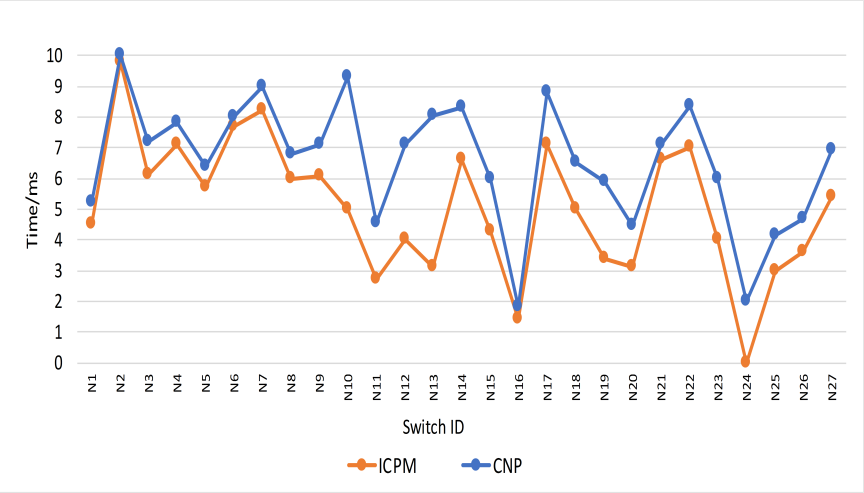

Figure 44: Failure recovery time: Norway Topology

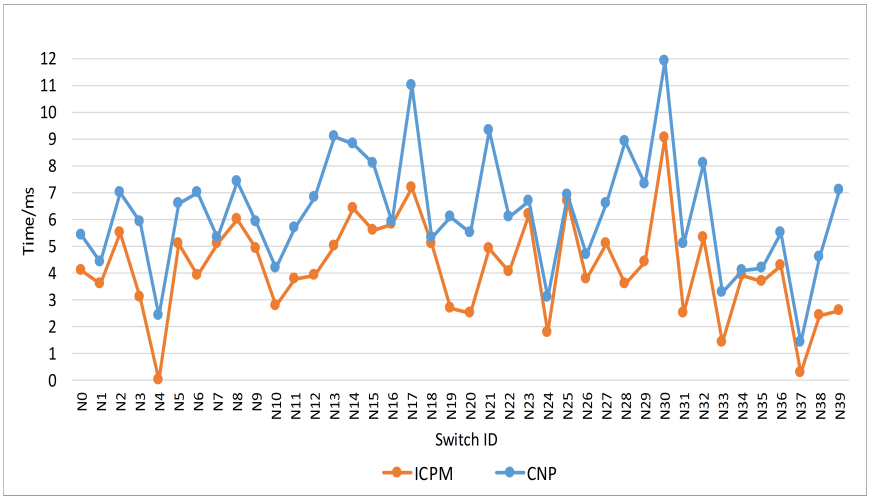

Figure 45: Failure recovery time: Pioro40 Topology

Figure 45 highlight the failure recovery time of Pioro40 topology. We can notice the difference in performance between our approach compared to CNP, as our solution demonstrates a better recovery time for all switches in the Pioro40 network.

5.6 Conclusions

The requirements of network survivability have been altered by SDN, where maintaining the connectivity between the controller and switches becomes a major challenge during disruptions in the network. In this sense, we investigate in-band control channel failure in SDN networks, where we propose a module approach to be integrated into the forwarding components in the network. A bypass and non-intervention backup recovery technique has been employed in this work to recover from switch failure in an SDN. Whenever a particular switch fails, the objective is to handle the switch failure, especially if it is related to a switch group-based. If the failure affects a group of dependent switches, the action is taken such that all control traffic of the affected switches is re-routed to an alternative recovery path. Significantly, this would help reduce the number of dropped packets that would occur during failure and minimize the recovery time of affected switches. Simulation results show that our solution has better performance in decreasing the packet dropping rate and recovery time.

Chapter 6

Towards Minimum Inter-Controller Delay Time in Software Defined Networking

In multiple SDN controller environments, the controllers need to maintain proper communication with other controllers. In this chapter, we give attention to the minimization issue of inter-controller delay time and delay time among forwarding elements in SDN, and determining the controllers' placement that satisfies these criteria.

6.1 Introduction

Today's networks have been improved through the introduction of new technologies aimed at fulfilling vital requirements that will enhance the networks' scalability, flexibility, and reliability. SDN is an emerging technology that provides a more inherent dependency relationship between the forwarding devices and the controller(s) to improve network performance and provide high-quality services. The main mechanism underlying SDN networks is the separation of the data and control planes from each other, making the network management process easier and less costly. At the same time, these improvements bring many challenges, especially in the control plane. Large-scale SDN networks are managed by multiple distributed controllers to avoid a single point of failure at the controller [13]. Yet, the major challenge remains as to how many controllers are appropriate and where these controllers should be placed.

To answer these two questions, various considerations and metrics can be used to find the best placement of controllers such as the delay time between the controllers and their associated switches,

the delay time between the controllers themselves, the load balance among the controllers, etc. Therefore, finding the optimal placement of controllers is a trade-off case based on the network's objectives.

When SDN is deployed in large-scale networks, they may consist of multiple domains, each with sets of switches managed by a single controller. Due to this distributed design, the controllers need to maintain proper communication with other controllers which should not be neglected [75] and as well as with its assigned switches. Most research works proposed in the literature give more attention to controller placements optimizing switch-controller communication, be it in terms of improving its latency, resilience, etc. On the other hand, the inter-controller communication has not received as much consideration, which is the main goal of this chapter. To this end, we apply the Connected Dominating Set (CDS) idea, which facilitates placing the controllers in such a way that it minimizes the delay time between the controllers while maintaining an acceptable delay time for controller-switch communication.

A dominating set (DS) is a subset S of all elements in the graph G such that each element in the graph G is either in S or is adjacent to at least one element in S. A connected dominating set (CDS) is a subset S of a graph G such that S forms a dominating set and S is connected [1]. The nodes in the CDS are called *dominators* and the rest of the nodes of the network which are one hop away from the CDS are called *dominatees* [76]. A CDS is a useful approach for routing and exchanging messages. From the SDN perspective, the controller placement problem can be simulated by a CDS architecture, where the CDS can helps to reduce the communication time between the controllers. The question of finding the CDS with minimum cardinality is called Minimum Connected Dominating Set (MCDS) which is an NP-Complete problem that requires heuristics to determine the CDS [77].

6.2 Related Work

SDN is centered on an innovation architecture of centralized control. This design is directed towards moving control plane functions from the network by decoupling the control and data planes from each other. The OpenFlow protocol [10] is the most common communication interface between a controller and switches in SDN. Therefore, in an OpenFlow architecture, a logically centralized controller is in charge of managing a set of switches by instructing them with routing rules that dictate their packet handling behavior. In SDN, a single, centralized controller cannot manage a large-scale network due to high traffic demands, which lead to a Single Point of Failure problem (SPoF) [78] with the controller failing. As a result, the deployment of multiple controllers is inevitable which in turn requires state synchronization between the controllers in order to maintain a consistent global

view of the network [19]. Since the controllers are required to communicate with their associated switches as well as conduct interacts with other controllers, the problem of controller placement is of particular importance as the location and capability of the controllers may significantly affect the performance of the network in terms of scalability, reliability, and availability.

The work conducted by Heller *et al.* [3] was the first mention of the controller placement problem in SDN. The authors concentrated only on minimizing the average and maximum (worst case) switch-to-controller latencies. At the same time, they ignored some other significant aspects, such as inter-controller latency and failure situations.

Other research proposed various strategies to cope with the multiple controller placement problem in SDN. Yeganeh and Ganjali [79] proposed a logically distributed controller framework called KANDOO, consisting of two layers of hierarchical design. The lower layer contains a set of local controllers without a network-wide view, where each controller manages a specific sub-domain. At the same time, the upper layer is a logically centralized (root) controller that maintains a network-wide view and manages all local controllers in the lower layer. KANDOO is a solution to the problem of massive traffic flow between local controllers, while it did not describe how the root controllers interact with each other to maintain a global view of the network, as well as the local controllers, to maintain scalability compared to a standard OpenFlow topology.

In multiple SDN controllers, maintaining a global view of the whole network is essential to ensure smooth network performance. However, inconsistent or stale states at the controller may result in the application layer making incorrect decisions, which then leads to inappropriate or sub-optimal operations of the network [19]. Yin *et al.* [80] propose "SDNi" as an East/Westbound interface between controllers in a distributed control environment. In SDNi, the network is split into multiple domains, each of which is being managed by a particular SDN controller. SDNi strives to facilitate exchanging the network status information among multiple domains and assisting in coordinating the controllers' decision-making processes.

Tanha *et al.* [47] propose a technique named Resilient Capacitated Controller Placement Problem (RCCPP) which considers different factors such as the capacity of the controllers, the load of the controllers, switch-controller propagation delay, and the propagation delay between controllers. Mouawad *et al.* [81] address the controller placement problem by focusing on the network load as the main factor and using a dynamic switch migration algorithm to minimize the switch-controller latency. They evaluate their scheme concerning the number of overloaded controllers and the number of migrated switches.

By dividing the entire network into multiple sub-networks, Qi *et al.* [82] introduces a modified density peaks clustering (MDPC) algorithm for planning the controllers' placement. Their goal is to minimize the average propagation delay time between switches and controllers, while they neglected

to examine the inter-controller delay time.

6.3 Network Model

We represent the network as a graph $G(V, E)$ where V are the graph nodes that denote the set of switches in the network, and E denotes the links set which are the connections between pairs of switches in V, the weight of each link is the propagation delay of the link. In SDN, the switches and controllers are the forwarding elements; therefore, we make the distinction in our work of assuming that the controllers' placements are at switch locations in the network. Controller-switches communication is constructed based on the in-band scheme [83]. Hence, there are no dedicated links between the controllers and their assigned switches, and all control messages are transmitted through the data link plane.

6.4 Proposed Algorithm

In this section, we present an algorithm for stable connected MCDS construction based on distance. The proposed method consists of three stages:

1. Constructing a Dominating Set (DS)
2. Building the Connected Dominating Set (CDS) that rely on the coloring method
3. Shrinking the connected dominating set to create a Minimum Connected Dominating Set (MCDS)

More details are provided below.

6.4.1 Stage 1: Dominating Set Construction

The following steps are performed to define the dominating set:

1. Initially we set the white color for all nodes in the network.
2. The node i with color white with the highest number of connected neighbors is selected from the graph $G(V, E)$ and identified a first black node (this node is a *dominator* node). In the event of more than one node with the same highest connectivity degree, a node with a minimum total shortest distance to all nodes in the network is selected to break the tie.
3. All 1-hop connected nodes (neighbors) of node i that are colored white are colored gray (i.e., these are *dominatees*).

4. Steps 2-3 are repeated through several iterations until all nodes in the graph $G(V, E)$ are colored either as black or gray.

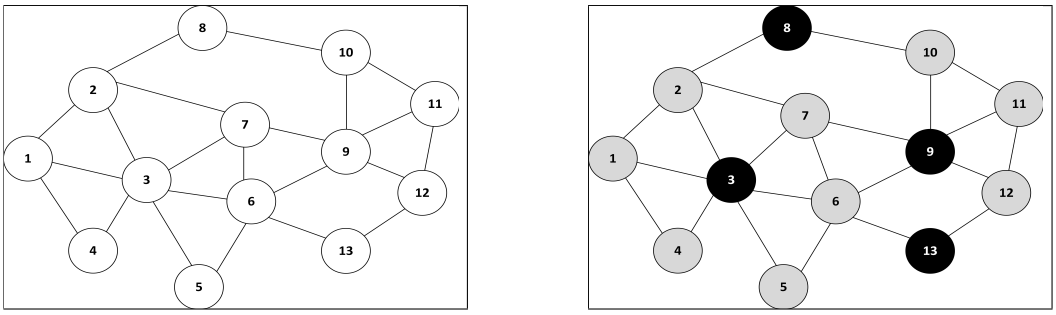

Figure 46: Stage 1: example of the Dominating Set (nodes colored black) selection process.

Stage 1 can be understood with the help of Figure 46. As seen in Figure 46, the left sub-figure shows all nodes initially are colored white. In the sub-figure on the right hand, node 3 is chosen as the first black node since it is the node associated with the largest number (6) of links. All its neighbors (1, 2, 4, 5, 6, and 7) are colored gray. The next choice is made on node 9, which has five links. Similarly, the color of all its connected neighbors (6, 7, 10, 11, and 12) are set to gray (note: nodes 6 and 7 were colored previously by node 3). After applying Phase 1 to an SDN architecture, a black node denotes a controller while a gray node refers to a switch.

6.4.2 Stage 2: Connected Dominating Set Construction

This stage aims to connect all nodes in dominating set D through finding a set of connectors C. For CDS formation, the following steps are performed:

1. Determine k, the maximum number of black nodes connected to any gray node. Let $i = k$.

2. Consider all gray nodes which are connected to exactly i black nodes. These selected nodes are *connectors* and are colored maroon. As observed in Figure 47, node 6 is selected as a connector which is indicated by the maroon color.

3. If the i black nodes connected to a current connector are all already connected to another connector, the current connector is deselected.
   As indictaed in Figure 47, node 12 is not selected as a connector since the black nodes it is connected to (nodes 9 and 13) are also connected to a connector (node 6).

4. If $i = 2$ and the black nodes are connected by more than one connector then the connector with less distance between the two black nodes will be chosen and rest are deselected. This is not done for $i > 2$ since it may result in a disconnected CDS.

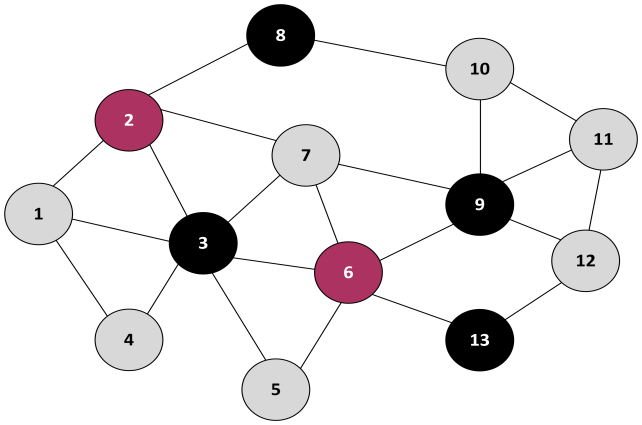

Figure 47: Determining Connectors

5. Check if the dominating set D gets connected; i.e. each black node has a direct link to another black node or connector.

6. If D gets connected or $i = 2$, continue to next step. Otherwise, the algorithm will go back to step 2 with $i = i - 1$.

7. Set the color of all connector nodes as black.

6.4.3 Stage 3: Minimum Connected Dominating Set Construction

The major objective of this stage is to minimize the size of the CDS obtained in stage 2. Determining the minimum appropriate number of the black nodes to meet the minimum delay time is a significant matter.

For MCDS formation, the following steps are performed:

1. We apply a single-iterate condition to convert back to being a switch (i.e., color as gray) any black nodes with only one link to another black node. The reason behind this is to maintain as few a number as possible of black nodes connected to the gray nodes within 1-hop or 2-hops. In this way, we ensure that the delay time between the controllers is a minimum while maintaining an acceptable delay time between the controllers and the switches.

2. Finally, we appoint each gray node (switch) to the closest black node (controller) forming multiple clusters. See Figure 48 where each resulting cluster is denoted by a different color (the controller of the cluster slightly darker in color).

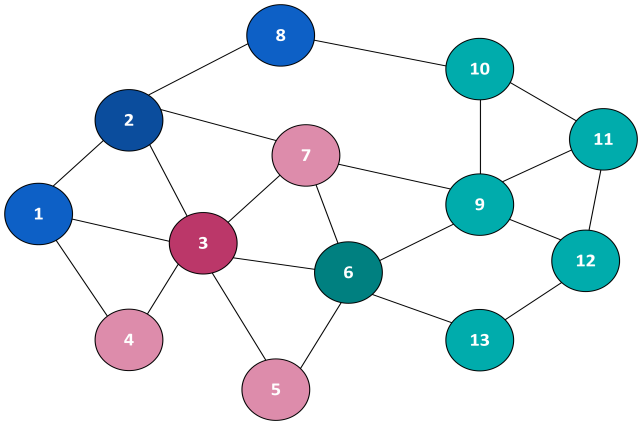

Figure 48: Final clusters

6.5 Performance Evaluation

6.5.1 Experimental Environment

The simulation experiments were conducted using `C++` and carried out on an Intel(R) Core(TM) i7-6770 CPU @3.40GHz and 16GB RAM with Windows 7 Pro (64-bit) operation system. The essential characteristics of the topologies used in the simulations are described in Table 7. Since we are dealing with real networks where every node has longitude and latitude, we employ the A* algorithm with the Haversine formula [84] to calculate the shortest distance between all node pairs in the network.

Network topology	Number of nodes	Number of links
ATT North America	25	57
India35	35	80
Bell Canada	48	65
Germany50	50	88

Table 7: Main characteristics of experimental topologies

In the following experiments, we compare our proposed approach with the HDIDS algorithm developed in [70] which uses an Independent Dominating set to determine controller placement.

6.5.2 Inter-Controller Delay Time

As a resilience metric, we investigate how the inter-controller delay time can be reduced based on the choice of controller placement, and what influence this has on the network performance. Formally, the inter-controller delay time for controller c_i is defined as the average of delay time $d(c_i, c_j)$ along

the shortest paths between c_i and all other controllers c_j in the set of controllers C:

$$\frac{1}{(|C|-1)|} \sum_{c_j \in C, i \neq j} d(c_i, c_j) \tag{4}$$

Ideally, the controllers should be placed as close together as possible to minimize inter-controller communication costs. Therefore, we employ the MCDS as the basis of our strategy to achieve the minimum delay time between the controllers.

In this part of the experiments, we conduct several simulations on different real networks to study inter-controller delay time. The inter-controller delay time varies a great deal from one controller to another. The reason is due to the formation process of the connected dominating set. In other words, the selection of dominators and connectors concerning the topology and geometry of the network leads to a wide range of shortest path lengths between controllers which in turn determines inter-controller delay time.

Figures 49–52 show the inter-controller delay time for four different networks. Figure 49 shows the inter-controller delay time between the 7 controllers determined by our algorithm for the India35 topology. We can observe that the inter-controller delay time of each controller separately for our algorithm is less compared to HDIDS, even with the significant variance in delay times. Further, the average inter-controller delay time across all controllers is 5.67ms for our approach compared to 8.43ms achieved by HDIDS.

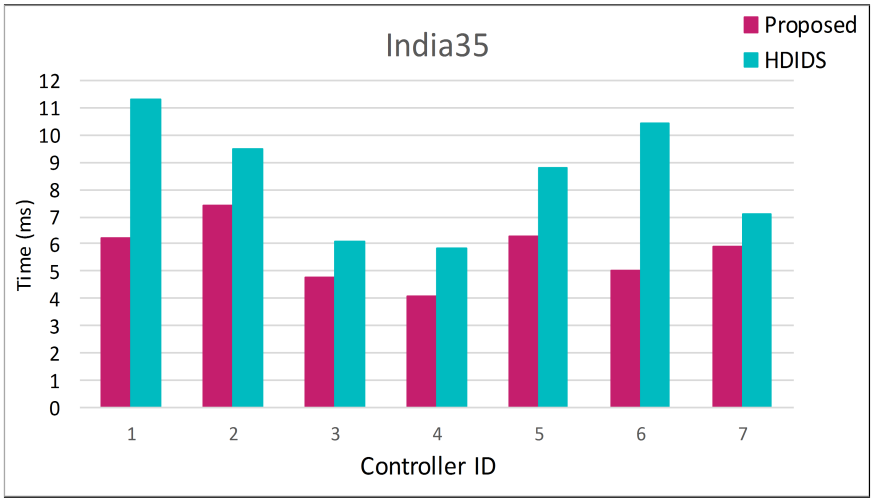

Figure 49: Inter-Controller Delay Time of India35 topology

Similarly, Figure 50 illustrates the results for the ATT topology, where the delay time is again less for our method as compared to HDIDS. The overall average inter-delay time by our proposed approach is 3.234ms, whereas 6.484ms by HDIDS.

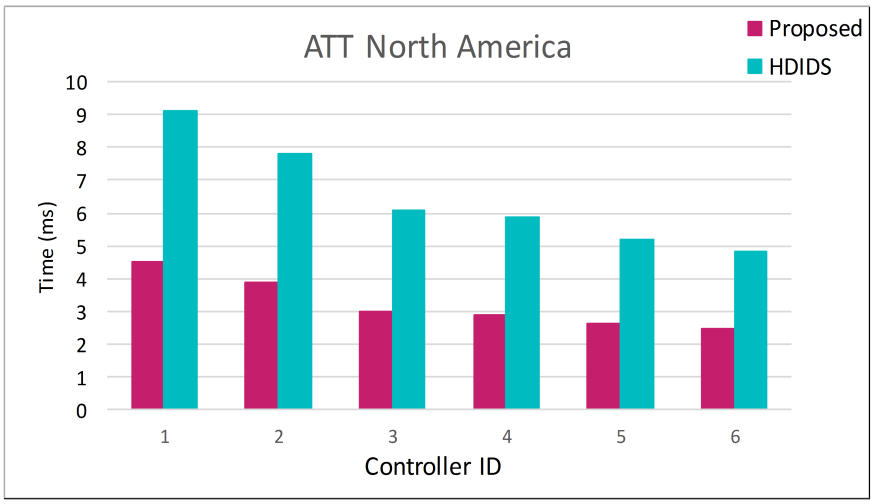

Figure 50: Inter-Controller Delay Time of ATT North America topology

In Figure 51, we determined 13 controllers with an overall average inter-controller delay time of 6.43ms compared to 8.811ms by HDIDS.

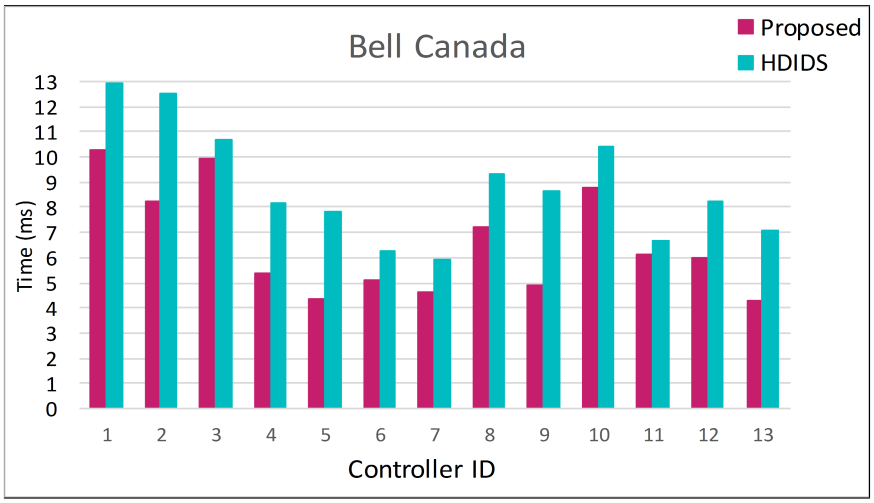

Figure 51: Inter-Controller Delay Time of Bell Canada topology

In Figure 52, German50 topology, our technique outperforms with an average inter-controller delay time of 6.69ms compared to 7.57ms by HDIDS. Note that in Figures 51 and 52, the inter-controller delay time for every controller is less than that determined by HDIDS (except for one controller for Germany50). The deployment of each type of SDN controller has a cost ($) which is based on different criteria (e.g., number of physical ports, the maximum number of requests that can be processed per second, etc.) [85].

As seen across all four topologies, we can realize that in small topologies, like India35 and ATT

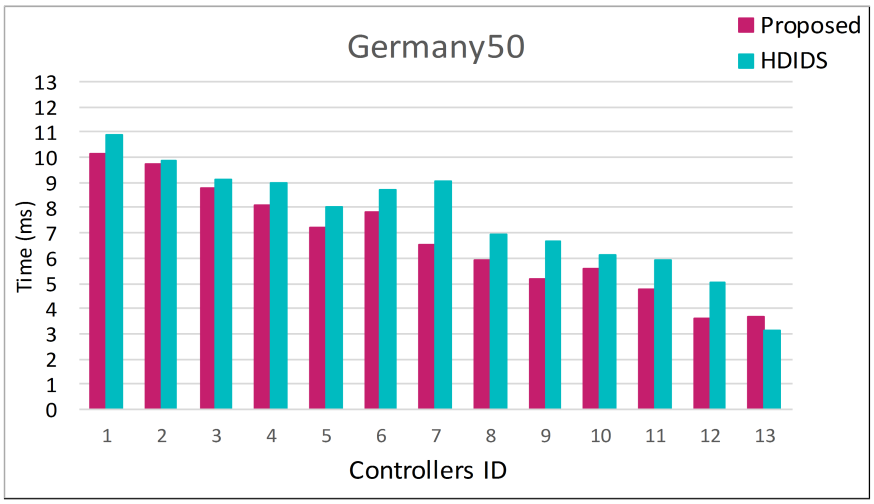

Figure 52: Inter-Controller Delay Time of Germany50 topology

North America, the number of controllers ranging between 6-7 have an average inter-controller delay time between 3-7ms.

In comparison, for slightly larger topologies (Bell Canada and Germany50), the controller number is around 13, while the inter-controller delay time goes down to 6.5-7ms. Thus, there is a trade-off between achieving the minimum delay time compared to the cost of the SDN controller deployment (i.e., number of controllers).

6.5.3 Required Number of Controllers

In order to demonstrate the extent and effectiveness of our method to place the controllers in SDN networks compared to HDIDS. Table 8 shows the required number of controllers obtained by our approach against HDIDS approach.

Network topology	Proposed	HDIDS
ATT North America	6	8
India35	7	8
Bell Canada	13	16
Germany50	13	14

Table 8: A comparison of the number of controllers.

From the Table 8, the number of controllers required to manage each network which obtained by our new technique is better than what was achieved previously.

6.5.4 Controller-Switch Delay Time

Even though the main objective of this work is to reduce the communication delay between the controllers, however maintaining the average controller-switch delay time within an adequate range is also desirable. To quantify the delay times in a network, we take the average of the delay times of the shortest paths from all switches in $S = V \backslash C$ to the controller closest to that switch which we denote as the controller–switch delay time as follow:

$$\frac{1}{|S|} \sum_{s \in S} \min_{c \in C} \; d(s, c). \tag{5}$$

For our analysis for controller-switch delay time, we will investigate using a range of the number of controllers. This analysis will be done by varying the single-iterate condition of stage 3 by allowing for a variable number of controllers to be removed from the CDS and making them switches. Note that this differs with our proposed strategy where the number of controllers is defined automatically to solve the controller placement problem as to study the inter-controller delay time in previous subsections.

Generally, to get less or much closer average controller-switch delay time, the number of controllers in the network must be increased. We present a comparison of results shown in Figures 53 to 56 for average controller-switch delay time between our proposed technique with a variable number of controllers and that of HDIDS. Note that x-axis shows the number of controllers, while the required number of controllers to obtain efficient inter-controller delay times, as discussed in Section 6.5.3, is highlighted by the red circles.

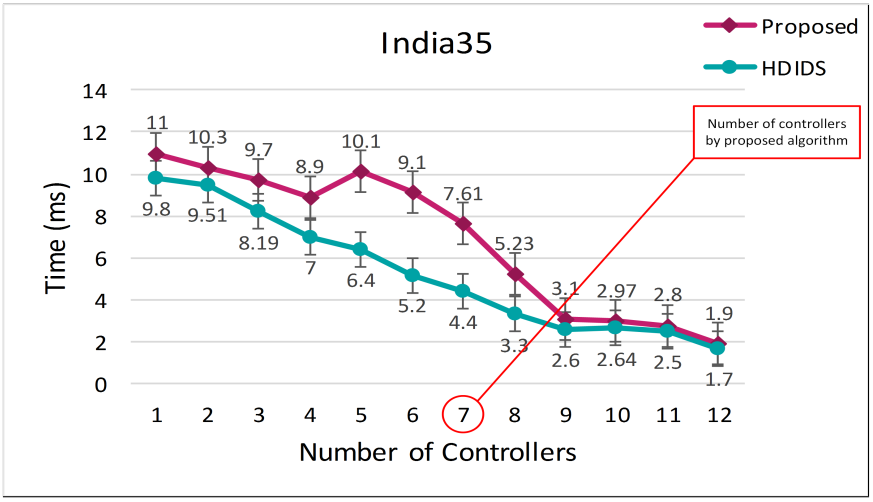

Figure 53: Average Controller-Switch delay time of India35 topology

According to Figure 53, at the beginning with one controller, HDIDS accomplishes an average

controller-switch delay time that is less than the proposed strategy. This relative comparison remains the case even with the continuation of the controller selection stages with a larger number of controllers. Our algorithm's average delay time jumps from 8.9 ms at four controllers to 10.1ms with five controllers. At seven controllers, as selected for inter-controller delay time in the previous sub-sections, our approach has an average controller-switch delay time of 6.89ms compared to 5.27ms for HDIDS. Then, gradually it begins to descend until it stabilizes to a very close level to HDIDS at nine controllers and higher.

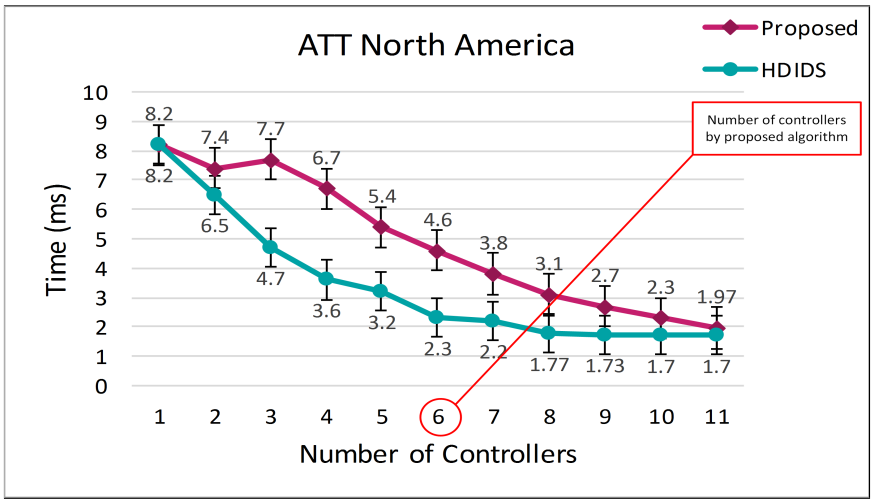

Figure 54: Average Controller-Switch delay time of Att topology

Regarding ATT topology, a similar scenario has been noted. At the three controllers, the average delay time rises from 7.4ms to 7.7ms at the next controller selection.

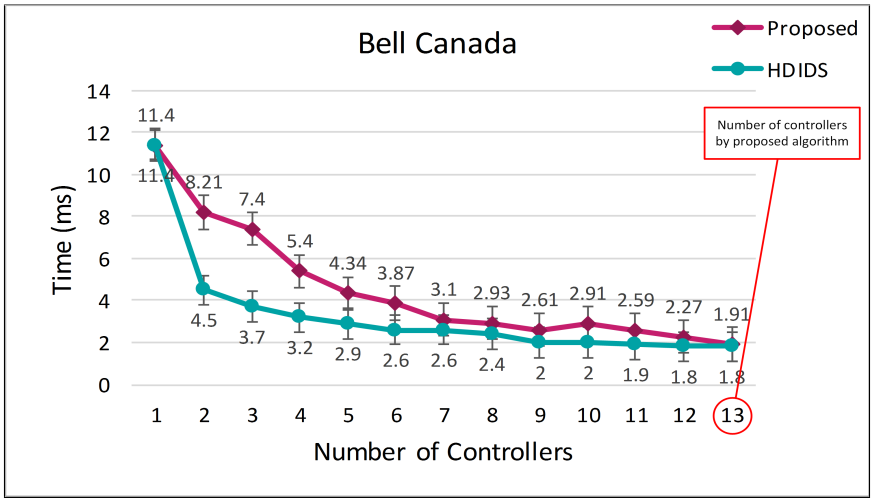

Figure 55: Average Controller-Switch delay time of Bell Canada topology

Figure 55 shows that our strategy follows the same behavior with Bell Canada topology. We notice that the delay time for our approach and the comparative approach started at 11.4ms at the first controller. At the second controller, the delay time for both approaches decreased, but at a different rate, as it decreased from 11.4 to 8.21ms in our approach, while it decreased further to 4.5ms in the HDIDS approach.

It can be seen that, according to our approach in the first and second topologies, there was a jump period between the controllers (i.e., between the controller 4 and 5 in the Figure 53, and between the controller 2 and the 3 in the Figure 54). But it was not repeated in the following topologies. Furthermore, in the first three topologies (Figures 53 to 55), the average delay time achieved by our approach at the maximum number of controllers is almost matched to HDIDS.

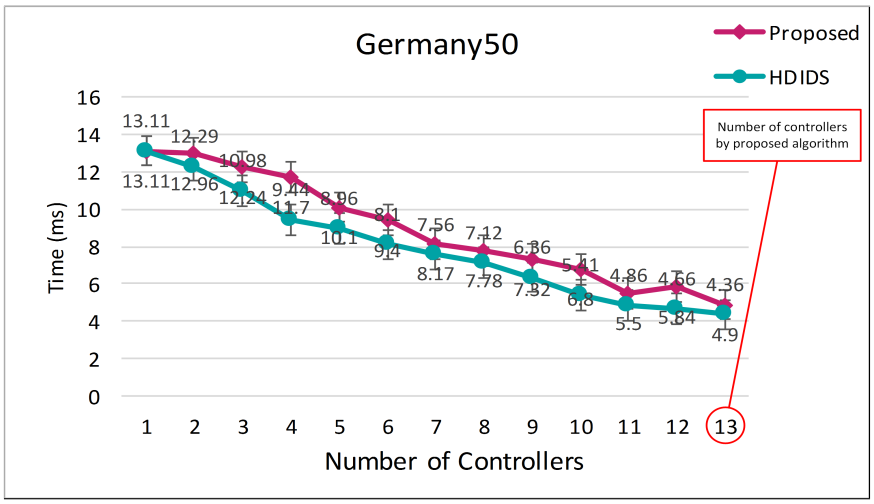

Figure 56: Average Controller-Switch delay time of Germany50 topology

As illustrated in Figure 56, in Germany50 topology, with the last found controller (13 controllers), the controller-switch delay time is 4.9ms compared to 4.36ms by HDIDS.

6.6 Conclusions

In this chapter, we proposed a new technique based on connected dominant set CDS. We aimed to determine the number of controllers and their placement with the goal of minimizing the inter-controller delay time in SDN networks. Our experimental results showed that our proposed strategy outperforms effectively compared to HDIDS. Our approach has achieved better results in terms of reducing the delay time between the controllers while maintaining nearly the same delay time between the controller and the switches compared to the HDIDS approach.

Even though the CDS strategy is an efficient approach in order to diminish the communication cost between SDN controllers, some aspects still require further investigation to gain the best outcomes. For instance, in the event of a controller's failure, all the switches associated with it are affected, and consequently, they are migrated to another controller. Therefore, the delay time between the affected nodes and the new controller is significant, especially for those switches located away from the new controller. As future work, we strive to study this scenario and find an adequate method to deploy the controllers in SDN networks considering the failure-based scenario and switch migration solution. Moreover, we conducted a single comparison of our approach with HDIDS, since HDIDS is the only approach that deals with the controllers' placement problem in SDN using dominating sets identification. Therefore, in terms of future work, we aim to future investigate the characteristics of our algorithms, and variants, by expanding the comparison of different metrics while including other vertex cover algorithms.

Chapter 7

Conclusions and Future Work

7.1 Summary

In large SDN networks, it is very hard to manage the entire network with one controller unit due to the increased number of forwarding nodes and increased number of routes that must be managed as well. Geographically, with larger SDNs, there is increased lag between the forwarding nodes and their controller. As a result, the controller units are no longer able to provide optimal network performance or QoS. Hence, using a certain number of controllers to handle this issue is desirable. In this thesis, we investigate the problem of the controller placement in SDN, and we study different related-aspects such as resilience, availability, scalability, and the control connection scheme as well. Various approaches have been proposed to tackle these issues. In the beginning, we propose a new technique named HDIDS to deal with the multi-controller placement problem with regards to minimizing the average and worst response time between the controllers and their assigned forwarding nodes.

Our approach in HDIDS is based on selecting the available controller with the highest connection degree and then exploiting the independent dominating set method to automatically form clusters of forwarding nodes associated with each controller in the network. We conducted experiments under many different metrics and topologies sized from medium to large. Compared with previously published work, experimental results showed that our approach provides better results, especially in large networks, in minimizing response time and maximal response time reduction as well.

As networks grow today, managing networks with a single controller is very hard due to the increasing number of switches that need a large number of paths through which data is routed to all components of the network. On one hand, increasing the number of forwarding nodes may force some of them to stay away from the control unit and thus increase the communication and response

time between the switches and the controller. As a result, the controller is no longer able to provide optimal network performance or Quality of Service. On the other hand, preserving a network of a large number of switches from the failure requires an efficient mechanism to deal with the placement of primary controllers and at the same time make network controllers robust enough to handle sudden failures, maintain availability within a permissible time frame. In this context and to improve the network's trustworthiness and its robustness under failure circumstances and ensure the stability of the network. We proposed a survivable backup controller placement approach that enhances the performance and throughput of the network. We evaluated our algorithms comprehensively by conducting several simulation practices using real topologies.

The outcomes showed that our backup strategy improved the network availability principle and reduced network breakdown opportunities. The requirements of network survivability have been altered by SDN, where maintaining the connectivity between the controller and switches becomes a major challenge during disruptions in the network. In this sense, as the third stage of this thesis, we investigate in-band control channel failure in SDN networks, and we propose a module approach be integrated into forwarding components in the network. A bypass and non-intervention backup recovery technique has been employed in this work to recover from switch failure in an SDN. Whenever a particular switch fails, the objective is to handle the switch failure especially if it is related to a switch group-based. If the failure affects a group of dependent switches, the action is taken such that all control traffic of the affected switches is re-routed to an alternative recovery path. Significantly, this helped in reducing the amount of dropped packets that would occur during failure and minimize the recovery time of affected switches.

Simulation results show that our solution has better performance in decreasing the packet dropping rate and recovery time. The inter-controller delay time is a significant part to be taking into account in SDN multiple controllers' environment construction. Therefore, in the last part of the thesis, we conducted a study on the problem of controller placement by minimizing the inter-controller delay time and the delay time among the switches. We proposed a new technique based on connected dominant set CDS aiming to determine the number of controllers and their placement within the SDN networks. Our experiment results showed that our proposed technique achieved better results in terms of reducing the delay time between the controllers while maintaining nearly the same delay time between the controller and the switches.

7.2 Future Work

The problem of SDN controller placement has several aspects that still need to be understood and requires further investigation. Different aspects are varied depends on the objective of the network. Regarding future work and directions beyond this research, in the following, we highlight a few interesting challenges that concern SDN:

7.2.1 Multiple controllers Failure

In a multi-controller SDN architecture, the controller plays a major role in the management and coordination of all components in the network. Therefore, the effect of the failure of one controller may be addressed and managed, unlike the failure of more than one controller simultaneously for any reason that may expose the entire network to collapse.
Hence, the study of this aspect needs to take into account all the possibilities that may cause this type of failure.

7.2.2 Switch Migration under Failure Event

Although, multiple distributed controllers are a promising approach to solve the problem of single controller failure. However, with the increasing network complexity, the switches' migration should be improved efficiency to enhance network performance under the failure scenarios. Different mechanisms to deal with the switches migration have been proposed. Yet, learning how to improve migration efficiency remains a difficult problem that needs more investigation.

7.2.3 Bypass-hop on multiple SDN control environments

The bypass-hop strategy is one of the efficient ways to tackle alternative pathfinding in the SDN networks. However, this mechanism can provide more reliable and trustworthy networks especially within dynamic SDN networks where the traffic pattern changes over time.

7.2.4 In-band and Out-of-band combination

Selecting the best control plane strategy in SDN networks still a key challenge, it requires efficient decision-making by taking into account several key considerations such as security, failure, and network bootstrapping time. The incorporation of in-band and out-of-band control scheme in SDN architecture can help to improve the network throughput and performance, and it should be handled as a multi-objective optimization model to solve this problem.

Bibliography

[1] D. Cokuslu, K. Erciyes, and O. Dagdeviren, "A dominating set based clustering algorithm for mobile ad hoc networks," in *International Conference on Computational Science*, pp. 571–578, Springer, 2006.

[2] S. Sezer, S. Scott-Hayward, P.-K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 36–43, 2013.

[3] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 7–12, ACM, 2012.

[4] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A K-means-based network partition algorithm for controller placement in software defined network," in *IEEE Intern'l Conf. on Communications (ICC)*, pp. 1–6, IEEE, 2016.

[5] M. Mendonca, B. N. Astuto, K. Obraczka, and T. Turletti, "Software defined networking for heterogeneous networks," 2013.

[6] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-defined networking (sdn): Layers and architecture terminology," in *RFC 7426*, IRTF, 2015.

[7] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 493–512, 2013.

[8] P. Neves, R. Calé, M. R. Costa, C. Parada, B. Parreira, J. Alcaraz-Calero, Q. Wang, J. Nightingale, E. Chirivella-Perez, W. Jiang, *et al.*, "The selfnet approach for autonomic management in an nfv/sdn networking paradigm," *International Journal of Distributed Sensor Networks*, vol. 12, no. 2, p. 2897479, 2016.

[9] N. Beheshti and Y. Zhang, "Fast failover for control traffic in software-defined networks," in *2012 IEEE Global Communications Conference (GLOBECOM)*, pp. 2665–2670, IEEE, 2012.

[10] ONF, "OpenFlow switch specification," 2013. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf.

[11] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, vol. 3, 2010.

[12] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, *et al.*, "Onix: A distributed control platform for large-scale production networks.," in *OSDI*, vol. 10, pp. 1–6, 2010.

[13] P. D. Bhole and D. D. Puri, "Distributed hierarchical control plane of software defined networking," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 516–522, IEEE, 2015.

[14] A. Capone, C. Cascone, A. Q. Nguyen, and B. Sanso, "Detour planning for fast and reliable failure recovery in SDN with OpenState," in *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, pp. 25–32, IEEE, 2015.

[15] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Automatic bootstrapping of openflow networks," in *2013 19th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, pp. 1–6, IEEE, 2013.

[16] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards sdn," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, pp. 363–368, IEEE, 2015.

[17] B. P. R. Killi and S. V. Rao, "Controller placement with planning for failures in software defined networks," in *IEEE Intern'l Conf. on Advanced Networks & Telecommun. Systems (ANTS)*, pp. 1–6, IEEE, 2016.

[18] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

[19] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 1–6, ACM, 2012.

[20] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 514–527, 2017.

[21] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339–1342, 2014.

[22] P. Xiao, W. Qu, H. Qi, Z. Li, and Y. Xu, "The SDN controller placement problem for WAN," in *IEEE/CIC International Conference on Communications in China (ICCC)*, 2014.

[23] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in SDN," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pp. 481–487, IEEE, 2016.

[24] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[25] G. Li, X. Wang, and Z. Zhang, "Sdn-based load balancing scheme for multi-controller deployment," *IEEE Access*, vol. 7, pp. 39612–39622, 2019.

[26] H. K. Rath, V. Revoori, S. Nadaf, and A. Simha, "Optimal controller placement in software defined networks (SDN) using a non-zero-sum game," in *IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6, IEEE, 2014.

[27] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE communications letters*, vol. 19, no. 1, pp. 30–33, 2015.

[28] V. Huang, G. Chen, P. Zhang, H. Li, C. Hu, T. Pan, and Q. Fu, "A scalable approach to sdn control plane management: High utilization comes with low latency," *IEEE Transactions on Network and Service Management*, 2020.

[29] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," in *25th Intern'l Teletraffic Congress (ITC)*, pp. 1–9, IEEE, 2013.

[30] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev, and P. Tran-Gia, "Specialized heuristics for the controller placement problem in large scale SDN networks," in *27th Intern'l Teletraffic Congress (ITC 27)*, pp. 210–218, IEEE, 2015.

[31] H. Selvi, G. Gür, and F. Alagöz, "Cooperative load balancing for hierarchical SDN controllers," in *IEEE 17th Intern'l Conference on High Performance Switching and Routing (HPSR)*, pp. 100–105, IEEE, 2016.

[32] S. Liu, H. Wang, S. Yi, and F. Zhu, "NCPSO: a solution of the controller placement problem in software defined networks," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 213–225, Springer, 2015.

[33] A. K. Singh and S. Srivastava, "A survey and classification of controller placement problem in sdn," *Int J Network Mgmt*, vol. 2018, p. 0, 2018.

[34] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.

[35] C. de Jaenisch, "Applications de l'analyse mathematique an jenudes echecs," 1862.

[36] C. Berge, "The theory of graphs and its applications, methuen & co," *Ltd., London*, 1962.

[37] O. Ore, *Theory of graphs*, vol. 38. American Mathematical Society, 1962.

[38] N. R. Chopde and M. Nichat, "Landmark based shortest path detection by using A* and Haversine formula," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 298–302, 2013.

[39] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[40] N. L. van Adrichem, F. Iqbal, and F. A. Kuipers, "Backup rules in software-defined networks," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 179–185, IEEE, 2016.

[41] D. Li, S. Wang, K. Zhu, and S. Xia, "A survey of network update in SDN," *Frontiers of Computer Science*, vol. 11, no. 1, pp. 4–12, 2017.

[42] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: An enhanced controller placement strategy for improving SDN survivability," in *2014 IEEE Global Communications Conference*, pp. 1909–1915, IEEE, 2014.

[43] L. Zhang, Y. Wang, W. Li, X. Qiu, and Q. Zhong, "A survivability-based backup approach for controllers in multi-controller SDN against failures," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 100–105, IEEE, 2017.

[44] F. A. Botelho, F. M. V. Ramos, D. Kreutz, and A. N. Bessani, "On the feasibility of a consistent and fault-tolerant data store for SDNs.," *EWSDN*, vol. 13, pp. 10–11, 2013.

[45] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 253–259, IEEE, 2016.

[46] S. Yang, L. Cui, Z. Chen, and W. Xiao, "An efficient approach to robust sdn controller placement for security," *IEEE Transactions on Network and Service Management*, 2020.

[47] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, 2018.

[48] B. P. R. Killi and S. V. Rao, "Towards improving resilience of controller placement with minimum backup capacity in software defined networks," *Computer Networks*, vol. 149, pp. 102–114, 2019.

[49] T. Hu, P. Yi, J. Zhang, and J. Lan, "Reliable and load balance-aware multi-controller deployment in sdn," *China Communications*, vol. 15, no. 11, pp. 184–198, 2018.

[50] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–2, IEEE, 2013.

[51] E. L. Lawler, J. K. Lenstra, A. R. Kan, D. B. Shmoys, *et al.*, *The traveling salesman problem: a guided tour of combinatorial optimization*, vol. 3. Wiley New York, 1985.

[52] A. Asadujjaman, E. Rojas, M. S. Alam, and S. Majumdar, "Fast control channel recovery for resilient in-band openflow networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pp. 19–27, IEEE, 2018.

[53] Y.-N. Bai, N. Huang, L.-N. Sun, and Y. Zhang, "Failure propagation of dependency networks with recovery mechanism," in *2017 Annual Reliability and Maintainability Symposium (RAMS)*, pp. 1–6, IEEE, 2017.

[54] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Fast failure recovery for in-band OpenFlow networks," in *2013 9th international conference on the Design of reliable communication networks (DRCN)*, pp. 52–59, IEEE, 2013.

[55] Y. Hu, W. Wendong, G. Xiangyang, C. H. Liu, X. Que, and S. Cheng, "Control traffic protection in software-defined networks," in *2014 IEEE Global Communications Conference*, pp. 1878–1883, IEEE, 2014.

[56] C.-Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid SDN networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1086–1094, IEEE, 2015.

[57] B. Niven-Jenkins, D. Brungard, M. Betts, N. Sprecher, and S. Ueno, "Requirements of an MPLS transport profile," tech. rep., RFC 5654, September, 2009.

[58] P. Thorat, R. Challa, S. M. Raza, D. S. Kim, and H. Choo, "Proactive failure recovery scheme for data traffic in software defined networks," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pp. 219–225, IEEE, 2016.

[59] C. Huang, V. Sharma, K. Owens, and S. Makam, "Building reliable mpls networks using a path protection mechanism," *IEEE Communications Magazine*, vol. 40, no. 3, pp. 156–162, 2002.

[60] S. El Rouayheb, A. Sprintson, and C. Georghiades, "Robust network codes for unicast connections: A case study," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 3, pp. 644–656, 2011.

[61] Y. Yu, L. Xin, C. Shanzhi, and W. Yan, "A framework of using openflow to handle transient link failure," in *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, pp. 2050–2053, IEEE, 2011.

[62] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Openflow-based segment protection in ethernet networks," *Journal of Optical Communications and Networking*, vol. 5, no. 9, pp. 1066–1075, 2013.

[63] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in openflow networks," in *2011 8th International Workshop on the Design of Reliable Communication Networks (DRCN)*, pp. 164–171, IEEE, 2011.

[64] H. Kim, M. Schlansker, J. R. Santos, J. Tourrilhes, Y. Turner, and N. Feamster, "Coronet: Fault tolerance for software defined networks," in *2012 20th IEEE international conference on network protocols (ICNP)*, pp. 1–2, IEEE, 2012.

[65] L. Ochoa-Aday, C. Cervelló-Pastor, and A. Fernández-Fernández, "Self-healing and SDN: bridging the gap," *Digital Communications and Networks*, 2019.

[66] Y. Park, D. T. Nguyen, B. Kang, K. Lee, J. Lee, and H. Choo, "A fast recovery scheme based on detour planning for in-band OpenFlow networks," in *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, pp. 1–5, 2017.

[67] K.-Y. Chan, C.-H. Chen, Y.-H. Chen, Y.-J. Tsai, S. S. Lee, and C.-S. Wu, "Fast failure recovery for in-band controlled multi-controller OpenFlow networks," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 396–401, IEEE, 2018.

[68] P. Goltsmann, M. Zitterbart, A. Hecker, and R. Bless, "Towards a resilient in-band SDN control channel," tech. rep., Universität Tübingen, 2017.

[69] C. Cascone, L. Pollini, D. Sanvito, A. Capone, and B. Sanso, "SPIDER: Fault resilient SDN pipeline with recovery delay guarantees," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pp. 296–302, IEEE, 2016.

[70] A. Alowa and T. Fevens, "Combined degree-based with independent dominating set approach for controller placement problem in software defined networks," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 269–276, IEEE, 2019.

[71] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[72] M. Edalatmanesh, "Heuristics for the critical node detection problem in large complex networks," Master's thesis, Brock University, St. Catharines, Ontario, Canada, 2013.

[73] J. Hwang, E. Jang, W. Lee, J. Kim, and H. Kim, "Topology design optimization for improving fail-over performance in wired mesh network," *IEMEK Journal of Embedded Systems and Applications*, vol. 14, no. 4, pp. 165–175, 2019.

[74] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0?survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[75] T. Zhang, P. Giaccone, A. Bianco, and S. De Domenico, "The role of the inter-controller consensus in the placement of distributed SDN controllers," *Computer Communications*, vol. 113, pp. 1–13, 2017.

[76] D.-Z. Du and P.-J. Wan, *Connected dominating set: theory and applications*, vol. 77. Springer Science & Business Media, 2012.

[77] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," in *Annals of Discrete Mathematics*, vol. 48, pp. 165–177, Elsevier, 1991.

[78] K.-Y. Wang, S.-J. Kao, and M.-T. Kao, "An efficient load adjustment for balancing multiple controllers in reliable SDN systems," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, pp. 593–596, IEEE, 2018.

[79] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 19–24, 2012.

[80] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "SDNi: A message exchange protocol for software defined networks (SDNs) across multiple domains," *IETF draft, work in progress*, 2012.

[81] N. Mouawad, R. Naja, and S. Tohme, "Optimal and dynamic SDN controller placement," in *2018 International Conference on Computer and Applications (ICCA)*, pp. 1–9, IEEE, 2018.

[82] Y. Qi, D. Wang, W. Yao, H. Li, and Y. Cao, "Towards multi-controller placement for SDN based on density peaks clustering," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[83] L. Schiff, S. Schmid, and P. Kuznetsov, "In-band synchronization for distributed SDN control planes," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 1, pp. 37–43, 2016.

[84] N. A. M. Nordin, Z. A. Zaharudin, M. A. Maasar, and N. A. Nordin, "Finding shortest path of the ambulance routing: Interface of A∗ algorithm using C# programming," in *2012 IEEE Symposium on Humanities, Science and Engineering Research*, pp. 1569–1573, IEEE, 2012.

[85] A. Sallahi and M. St-Hilaire, "Expansion model for the controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 21, no. 2, pp. 274–277, 2016.