Flexible Architecture for the Future Internet Scalability of SDN Control Plane

Kurdman Abdulrahman Rasol

Advisor: Prof. Jordi Domingo-Pascual

Departament d'Arquitectura de Computadors (DAC)
Universitat Politècnica de Catalunya (UPC)

This dissertation is submitted for the degree of
Doctor of Philosophy in Computer Architecture

December 2021

Acknowledgements

This dissertation is the culmination of my work over the last four years. This entire period has been an extraordinarily beneficial experience for me, both academically and professionally. Throughout this incredible trip, I have met many people who have contributed–directly or indirectly–to my thesis. This is only a humble attempt to express my heartfelt gratitude to those who deserve special recognition for advising, supporting, collaborating, or just being there when I need them most.

First and foremost, my sincere gratitude goes to my PhD supervisor, Prof. Dr. Jordi Domingo-Pascual, for the opportunity to pursue my PhD under his guidance. I am grateful for his patience, motivation, and support over the past four years, which enabled me to push the boundaries of my limited knowledge and understanding of 5G networks, Software-Defined Networking, Network Function Virtualization, Controller Placement Problem, Network Architecture, Routing, Switching, the Future Internet, Computer Networks, and Communications. Prof. Domingo, who provided me with direction, technical support, and many encouraging conversations and fruitful discussions that helped me overcome the initial faltering steps I encountered, thank you so much!

Let me also acknowledge the great support I received from the members of the Broadband Communications Systems and Architectures (CBA) Research Group, as well as the Computer Architecture Department in general. My special thanks to the former coordinator of the Doctoral Program, Prof. Dr. Xavi Masip Bruin, for his firm support during the past four years in the CBA Research Group.

The work contained in this dissertation was partially supported by the Spanish Ministry of Economy and Competitiveness under contract TEC2017-90034-C2-1-R (ALLIANCE). This work has been also partially a part of the Spanish I+D+i project TRAINER-A (ref. PID2020-118011GB-C21), funded by MCIN/AEI/10.13039/501100011033. In addition, I appreciate the support from the Universitat Politecnica de Catalunya (UPC), which is gratefully acknowledged.

I am extremely satisfied with the results of the research I conducted with Prof. Dr. Jordi Domingo's support, and I appreciate how working under his supervision gave me an insight into the dynamics and the inner workings of different research groups. I also

wish to acknowledge the inestimable and thank my friends, colleagues from UPC, and Cost Action bodies in different positions: Pepa Bayari, Dejan Vicomej, Rashmi Bakshi, Ilker Demirkol, Albert López, José Rafael Suárez, Albert Mestres, Sergi Abadal, Jordi Paillissé, Paul Almasan, Hamidreza Taghvaee, and Ismael Castell. I am especially grateful for the many good friends I have made, both in Barcelona and abroad.

Last but not least, I am thankful for the invaluable support of my family and friends in the Kurdistan Region, Spain, and elsewhere, for making me put things in perspective and helping me keep a balanced life, keeping my worries at bay by getting my mind off the troubles in academia and research during leisure times spent together.

To my parents, the role models who provided me with the education, motivation, loving guidance, and unconditional support that has led me successfully through this journey, I am forever indebted. I am forever grateful to my devoted mother, Qumri Khaled, for her endless support, tremendous wisdom, guidance, integrity, selflessness, faithful prayers, and abounding and undying love. She has had the biggest impact on my life and I would never have achieved it here without her endless support and encouragement.

Finally, and most importantly, I'd like to thank my esteemed brother, Dr. Mezgeen Rasol, and my friend, Dr. Sarbast Ahmad, for everything we've shared over the last few years (for the love, encouragement, support, understanding, concern, great cooking, and a wicked sense of humor during the ups and downs of my doctoral studies).

Abstract

Software-Defined Networking (SDN) separates the control plane from the data plane. The initial SDN approach involves a single centralized controller, which may not scale properly as a network grows in size. Distributed controllers have emerged to address the disadvantages of a single centralized controller. The control architecture needs to be distributed with traffic control between switches and controllers and among the controllers in order to allow SDNs for several thousand switches. One of the most significant research challenges for distributed controller architectures is to effectively manage controllers, which includes allocating enough controllers to appropriate network locations. To address these daunting issues, we make the following major contributions:

This thesis expands the method of solving the Control Placement Problem (CPP) based on the K-means and K-center algorithms to include a Hierarchical Controller Placement Problem (HCPP), located at a high level of Super Controller (SC), a middle level of Master Controllers (MCs), and the lowest level of domain controllers (DCs). The optimization metric addresses latency between the controller and the switches assigned to it.. The proposed architecture and methodology are implemented using the topology of Western European NRENs from the Internet Topology Zoo. The entire network topology is divided into clusters, and the optimal number of controllers (DCs) and their placement are determined for each cluster. MC placement optimization determines the optimal number of MCs and their optimal placement.

As a second contribution, an accumulated latency is defined to solve CPP, which takes into account both the latency between the controller and its associated switches and the latency between controllers. Under the constraint of latency, an optimization problem is formulated as per mixed-integer linear programming (MILP). The goal of the research is to reduce accumulated latency while also reducing the number of network controllers and optimizing their placement to achieve an optimal balance. The performance of the developed method is evaluated on Internet2 OS3E real network topology.

To achieve the third objective, a metric was developed that includes reliability. The communication latency between controllers should also be considered because a low controller-switch delay does not always imply a short controller-controller delay for a particular

controller placement. As the third contribution, we propose a novel metric for CPP to improve the reliability of controllers that takes into account both communication latency and communication reliability between switches and controllers, as well as between controllers. When a single link fails, reliability is taken into account. This aspect concluded by identifying the optimal controller placement to achieve low latencies in control plane traffic. The goal of this project is to reduce the average latency.

As the fourth contribution, this study evaluates the Joint Latency and Reliability-aware Controller Placement (LRCP) optimization model. As the evaluation metric, control plane latency (CPL) is defined as the sum of the average switch-to-controller latency and average inter-controller latency. The latency of the control plane, utilizing the actual latencies of the real network topology, is calculated for every optimum placement in the network. In the case of a failure of the single link, the actual CPL for LRCP placements is calculated and evaluated to determine how good LRCP placements are. CPL metrics are used to compare latency and reliability metrics with other models.

This study provides proof that the developed methodologies for large-scale networks are highly powerful in terms of searching for all feasible controller placements while assessing the outcomes. In addition, compared to previous work including latency among controllers and reliability for an event of single-link failure.

Resum

La xarxa definida per programari (SDN) separa el pla de control del pla de dades. L'enfocament SDN inicial implica un únic controlador centralitzat, que pot no escalar correctament a mesura que la xarxa creixi de mida. Els controladors distribuïts han sorgit per abordar els inconvenients d'un únic controlador centralitzat. . Un dels reptes de recerca més importants per a les arquitectures de controladors distribuïts és gestionar de manera eficaç els controladors, que inclou l'assignació de controladors suficients a les ubicacions de xarxa adequades. Per abordar aquests problemes, fem les següents contribucions.

Aquesta tesi amplia el mètode de resolució del Problema de Col·locació de Control (CPP) basat en els algorismes de K-means i K-center per incloure un Problema de Col·locació de Controladors Jeràrquics (HCPP), situat a un nivell alt de Super Controller (SC), un nivell de controladors mestres (MC) i el nivell més baix de controladors de domini (DC). La mètrica d'optimització és la latència entre el controlador i els commutadors assignats a aquest. L'arquitectura i la metodologia proposades s'implementen utilitzant la topologia de NREN d'Europa occidental de l'Internet Topology Zoo. La topologia de la xarxa es divideix en clústers i es determina el nombre òptim de controladors de domini (DC) i la seva ubicació per a cada clúster. L'optimització de la ubicació de MC determina el nombre òptim de MC i la seva col·locació òptima.

Com a segona contribució, es defineix una latència acumulada per resoldre el CPP, que té en compte tant la latència entre el controlador i els seus commutadors associats com la latència entre controladors. Sota la restricció de la latència, es formula un problema d'optimització segons la programació lineal de nombres enters mixts (MILP). L'objectiu de la investigació és reduir la latència acumulada alhora que es redueix el nombre de controladors de xarxa i optimitza la seva col·locació per aconseguir un equilibri òptim. El rendiment del mètode desenvolupat s'avalua en la topologia de xarxa real d'Internet2 OS3E.

Per aconseguir el tercer objectiu, es va desenvolupar una mètrica que inclou la fiabilitat. També s'ha de tenir en compte la latència de comunicació entre controladors perquè un retard baix entre el commutador i el controlador no sempre implica un retard curt del controlador-controlador per a una ubicació concreta dels controladors. Com a tercera contribució, proposem una nova mètrica per al CPP per millorar la fiabilitat dels controladors que tingui

en compte tant la latència de la comunicació com la fiabilitat de la comunicació entre commutadors i controladors, així com entre controladors. La fiabilitat es té en compte quan falla un únic enllaç identificant la col·locació òptima dels controladors per aconseguir baixes latències en el trànsit del pla de control. L'objectiu d'aquest projecte és reduir la latència mitjana.

Com a quarta contribució, aquest estudi avalua el model d'optimització Joint Latency and Reliability-aware Controller Placement (LRCP). Com a mètrica d'avaluació, la latència del pla de control (CPL) es defineix com la suma de la latència mitjana de commutador a controlador i la latència mitjana entre controladors. La latència del pla de control, utilitzant les latències reals de la topologia de xarxa real, es calcula per a cada col·locació òptima a la xarxa. En el cas d'una fallida en un únicenllaç, es calcula i s'avalua el CPL real de les ubicacions LRCP per determinar com de bones són les ubicacions LRCP. Les mètriques CPL s'utilitzen per comparar les mètriques de latència i fiabilitat amb altres models.

Aquest estudi proporciona la prova que les metodologies desenvolupades per a xarxes a gran escala són molt potents pel que fa a la recerca de totes les ubicacions de controladors factibles mentre s'avaluen els resultats. A més, en comparació amb el treball anterior, inclou la latència entre els controladors i la fiabilitat per a un esdeveniment de fallada d'un enllaç únic.

Resumen

Las redes definidas por software (SDN) separan el plano de control del plano de datos. El enfoque inicial de SDN implica un único controlador centralizado, que puede no escalar adecuadamente a medida que una red crece en tamaño. Los controladores distribuidos han surgido para abordar las desventajas de un único controlador centralizado. Uno de los retos de investigación más importantes para las arquitecturas de controladores distribuidos es la gestión eficaz de los controladores, que incluye la asignación de suficientes controladores en las ubicaciones adecuadas. Para hacer frente a estos problemas, realizamos las siguientes contribuciones principales:

Esta tesis amplía el método de resolución del Problema de Colocación de Controles (CPP) basado en los algoritmos K-means y K-center para incluir un Problema de Colocación de Controladores Jerárquicos (HCPP), situado en un nivel alto de Super-controladores (SC), un nivel medio de Controladores Maestros (MC), y el nivel más bajo de controladores de dominio (DC). La métrica de optimización es la latencia entre el controlador y los conmutadores asignados al mismo. . La arquitectura y la metodología propuestas se implementan utilizando la topología de las NREN de Europa Occidental del TopologyZoo. La topología completa de la red se divide en clústeres, y se determina el número óptimo de controladores de dominio (CD) y su colocación para cada clúster. La optimización de la colocación de los MC determina el número óptimo de MC y su colocación óptima.

Como segunda contribución, se define una latencia acumulada para resolver el CPP, que tiene en cuenta tanto la latencia entre el controlador y sus conmutadores asociados como la latencia entre los controladores. Bajo la restricción de la latencia, se formula un problema de optimización según la programación lineal de enteros mixtos (MILP). El objetivo es reducir la latencia acumulada al tiempo que se reduce el número de controladores de la red y se optimiza su ubicación para lograr un equilibrio óptimo. El rendimiento del método desarrollado se evalúa en la topología de Internet2 OS3E.

Para lograr el tercer objetivo, se desarrolló una métrica que incluye la fiabilidad. La latencia de la comunicación entre controladores también debe tenerse en cuenta, ya que un bajo retardo entre controladores y conmutadores no siempre implica un corto retardo entre controladores para una determinada ubicación de los mismos. Como tercera contribución,

proponemos una nueva métrica para el CPP para mejorar la fiabilidad de los controladores que tiene en cuenta tanto la latencia de la comunicación como la fiabilidad de la comunicación entre los conmutadores y los controladores, así como entre los controladores. Se tiene en cuenta la fiabilidad cuando falla un solo enlace. Este aspecto concluye con la identificación de la ubicación óptima de los controladores para lograr bajas latencias en el tráfico del plano de control. El objetivo es reducir la latencia media.

Como cuarta contribución, este estudio evalúa el modelo de optimización Joint Latency and Reliability-aware Controller Placement (LRCP). Como métrica de evaluación, la latencia del plano de control (CPL) se define como la suma de la latencia media entre conmutadores y controladores y la latencia media entre controladores. La latencia del plano de control, utilizando las latencias reales de la topología de la red, se calcula para cada ubicación óptima en la red. En el caso de un fallo de un enlace, se calcula y evalúa la CPL real para las colocaciones de LRCP con el fin de determinar lo buenas que son las colocaciones de LRCP. Las métricas CPL se utilizan para comparar las métricas de latencia y fiabilidad con otros modelos.

Este estudio demuestra que las metodologías desarrolladas para redes a gran escala son muy potentes en cuanto a la búsqueda de todas las ubicaciones factibles de los controladores mientras se evalúan los resultados. Además, en comparación con los trabajos anteriores, que incluyen la latencia entre controladores y la fiabilidad para un caso de fallo de un solo enlace.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

1G First Generation

2G Second Generation

3G Third-Generation

4G Fourth Generation

5G Fifth Generation

A-CPI Application-Controller Plane Interface

ABFO Adaptive Bacterial Foraging Optimization

AHP Analytic Hierarchy Process (

AMPS Advanced Mobile Phone System

API Application programming interface

AS Autonomous System

BF Brute Force

BPL Backup paths latency

C-DPI Controller Data Plane Interfaces

CAPEX Capital expenditure

CC controller-to-controller communication / latency

CCI controller-to-controller interface

CCP Capacitated Controller Placement

CCPP Capacitated Controller Placement Problem

CDF Cumulative Distribution Function

CDMA Code MultipleAccess Division

CGLCPP Global Latency Control Placement Problem with Capacitated Controllers

CLACPM Capacity and Load-Aware SDN Controller Placement

CLEP Cluster Leader Election Problem

CLI Command-line interface

CLPA Capacitated Label Propagation Algorithm

CN Controller Number

CNCP Capacitated Next Controller Placement

CNM Control Network Mapping

CNPA Clustering-based Network Partition Algorithm

CO-FFCCP FFCCP With Combined Objective

CP Controller placement

CPAPLS Controller Placement Anytime Pareto Local Search

CPC capacitated p-center

CPCNS Controller Placement under Comprehensive Network States

CPGA Controller Placement Genetic Algorithm

CPL Control plane latency

CPP Controller Placement Problem

CPSLF Controller Placement under Single Link Failure

CS controller-to-switch communication / latency

D2D Device-to-Device communication

DBCP Density Based Controller Placement

DC Domain Controller

DC-DC Domain Controller and Domain Controller communication / latency

DC-S Domain Controller and Switch communication / latency

DCPP Dynamic Controller Provisioning Problem

DNS Domain name service

EA Evolutionary Algorithm

EDGE Enhanced DataRate for GSM Evolution

eMBB enhanced Mobile Broadband

ETSI European Telecommunications Standards Institute

EVDO Evolution-Data Optimized

FDMA Frequency division multiple access

FFA Firefly algorithm

FFCCP Failure Foresight Capacitated Controller Placement

FTCP Fault Tolerant Controller Placement

GA Greedy Algorithm

GC Graph Clustering

GPRS General Packet Radio Service

GROM Global Resilient Orchestration Modules

GSM Global Systems for Mobile communications

HCPM Heterogeneous Cost Placement Model

HCPP Hierarchical Controller Placement Problem

HCS Highly Connected Sub-graphs

HPP Hypervisor Placement Problem

HRA Heuristic Approach

HSDPA High Speed Downlink Packet Access

HSPA+ High Speed Packet Access

HSUPA High-Speed Uplink Packet Access

I-CPI Intermediate-Controller Plane Interface

IDS Intrusion detection systems

ILP Integer Linear Programming

IoT Internet of Things

IP Internet Protocol-based applications

IQP Integer Quadratic Programming

IS-95 Interim Standard 95

ITU International Telecommunication Union

JHCP Jointly Optimizing the Placement of Hypervisors and Controllers

LACP Link Aggregation Control Protocol

LARC Latency-Aware Reliable Controller Placement

LCP Joint Placement Latency Optimization

LFACCP Link Failure Aware Capacitated Controller Placement

LLDP Link Layer Discovery Protocol)

LRCP Joint Latency and Reliability-Aware Controller Placement

LROM Local Resilient Orchestration Modules

LTE-A Long-Term Evolution-Advanced

MC Master Controller

MC-DC Master Controller and Domain Controller communication / latency

MC-MC Master Controller and Master Controller communication / latency

MC-SCA Minimum Cost Switch-Controller Association

MCC A Min-cover based Controller Placement

MCC Mobile Cloud Computing

MCDA Multi-criteria Decision Algorithms

MCPS Multi-controller Placement Scheme

MCPS Multi-controller placement scheme

MDCP Measurement-aware Distributed Controller Placement

MHNSGA Multi-start hybrid non-dominated sorting genetic algorithm

MILP Mixed-integer linear programming

MIMO Multiple-input Multiple-output

MLkP Multilevel k-way Partition

mMTC Massive Machine-Type Communication

MOCP Multi-objective Optimization Controller Placement

MOGA Multi-Objective Genetic Algorithm

NAT Network address translation

NBI Northbound Interface

NCPSO Network Clustering-based Particle Swarm Optimization Algorithm

NFV Network Function Virtualization

NFV MANO NFV management and orchestration

NFVI NFV infrastructure

NMT Nordic Mobile Telephone

NREN National Research and Education Network

OKMP Optimized K-means network partition algorithm

ONF Open Networking Foundation

OpEx Operational expenditure

OSCP Opendaylight SDN Controller Platform

PAM Partitioning Around Medoids

POCO Pareto-based Optimal Controller Placement

PPL Primary paths latency

PSO Particle Swarm Optimization

QIP Quadratic Integer Programming

QoS Quality of Service

RCCPP Resilient Capacitated Controller Placement Problem

RCP Reliability-aware Controller Placement algorithm

RCP Resilient Controller Placement

RCP-DCP Disjoint Control Paths

RCP-DCR Different Controller Replicas

RCPP Reliable controller placement problem

REST Representational state transfer

RL Reliability-Latency

RMM-FB Resilient Multi-controller Mapping with Full Backup

RMM-LM Resilient Multi-controller Mapping with Latency Minimization

RMM-MB Resilient Multi-controller Mapping with Minimum Backup Capacity

RMM-MC Resilient Multi-controller Mapping with Minimum Cost

RRH Remote Radio Head

RSPAN Remote SPAN

SA Simulated Annealing

SBI Southbound Interface

SC Super Controller

SC-MC Super Controller and Master Controller communication / latency

SDN Software Defined Networking

SDWANs Software-Defined Wide Area Networks

sFlow Sampled flow

SLA Service level agreement

SLF Single-Link Failure

SMS Short Message Service

SPAN Switch port Analyzer

SPOF Single point of failure

SS Switch-switch communication / latency

TACS Total Access Communication System

TCP Transmission Control Protocol

TCS Tata Consultancy Services

TLBO Teaching-learning based optimization

UCPP Uncapacitated Controller Placement Problem

UDP User Datagram Protocol

UMTS Universal Mobile Telecommunications Systems

URLLC Ultra-Reliable and Low Latency Communication

VBO Varna-based optimization

VCPP Controller Placement in Virtual SDN

VN Virtual Network

VNF Virtual network functions

WAN Wide Area Network

WCDMA Wideband Code Division Multiple Access

WSDN Wireless Software Defined Networking

Chapter 1

Introduction

1.1 Motivation and Challenges

In traditional networks, both the control and data planes are embedded in the same network hardware or device. The control plan is responsible for configuring and programming paths that control information for data flow and forwarding. Because this mechanism is not dynamic in order to define information properly, experts cannot alter it in different configurations. This represents a challenge for the traditional networking approach in terms of reactivity to dynamic changes in network traffic issues. The number of configured network devices increases commensurately with the complexity of network deployment, especially in the context of Wide Area Network (WAN). The manual configuration of plentiful devices is error-prone and time-consuming, thus there is a need for more simple and automated solutions for the management of complex traditional networks.

Software Defined Networking (SDN) facilitates network operators to simplify network management by enabling scalability, error-free, dynamically configurable networks, reducing the amount of time required. It provides programmability to facilitate the configuration and administration of a network that is constantly changing its network state [2]. In Software Defined Networks, the data plane with switches is separated from the logical plane to make network management easier and more flexible, centralizing the logical plane where all decision-making occurs. Since all forwarding decisions depend on the central controller, all switches may misbehave in case of a failing controller and no longer facilitate the provision of network-based services [3], [4].

Moreover, the switch forwards data for the first time, finding the respective flow entry from the table. During the process, if the corresponding entry flow is not found in the system, it will request the most recent flow table from its controller. Controllers comprise the control plane, while the data plane comprises the switches. A typical SDN includes a controller that

handles routing decisions and switches that operate on the controller's decisions. However, this has constraints, because a single controller gets overloaded and becomes a Single Point of Failure (SPOF) occurring in the network. This reduces the network's scalability and performance [5].

The vast majority of research in this area has been focused on controller placement, since controller location plays a crucial role in determining the latency between switches and corresponding controllers. Basically, the controller can simply manage the entire network.

In the control plane, a single controller is used in a small-sized network with good performance for such networks. However, larger networks consume more time and have reduced performance because of large coverage area and long propagation, due to utilizing a single controller. Therefore, multiple controllers are required to be deployed on a large-sized network considering latency, reliability, scalability, and performance. The majority of the research is focused on the placement of multiple controllers rather than a single one, which can work concurrently to serve and provide the switches better.

However, this approach raises another problem. The placement of multiple controllers has become a big challenge due to the changing status of the network. Depending on the number of switches and hosts in a network, the number of controllers should be increased. This leads to a balance between the controllers. When the controllers balance the traffic across the network, the network performance increase can be seen in terms of increased throughput and reduced latency.

The controller placement is a major problem with multi-controllers in the control plane. Network devices use a variety of metrics to select number and placement of the controllers. As a result, in order to improve the performance of networks, there are three main questions regarding the controller placement problem:

1. How many controllers are required on the network?

2. Where should controllers be placed on the network?

3. Which metrics should be considered?

The best controller placement can be achieved by using different metrics and considerations based on the network's objectives. The number of controllers in a given network is another aspect of the controller placement. Deploying a certain number of controllers influences several parameters, such as latency and reliability.

Other issues include determining the best minimum number of controllers and deciding where they should be placed, given that the cost of deployment rises as the number of controllers increases; and controller traffic management and load imbalance – higher traffic

on the controller causes a degradation of network performance and network failure. Therefore, this study proposes strategies to address these issues in line with the above research questions.

There are many works of literature and proposals regarding the SDN distribution controllers in the control plane, but only a few of them considered the problem of controller placement (or allocation) for a given network topology with a certain number of controllers. In fact, the impact on service performance imposed by a controller's allocation has not been thoroughly studied. The paper by Heller et al. [6] is one of the first studies of the placement problem, and this study addresses some limitations raised by that paper.

Motivated by these facts, this thesis develops an approach based on [6]. However, instead of searching for the optimal controller placement only in terms of controller-to-switch, this approach takes into account a variety of factors, including both controller-controller and controller-switch traffic (both are measured as latency).

In distributed controllers, the control traffic is a combination of controller-to-switch (CS) and controller-to-controller (CC) traffic. The load of the CS traffic is shared among the controllers, but at the expense of additional traffic for the CC communication. The inter-controller traffic is necessary for the synchronization of the controllers. Distributed controllers use coordination protocols and algorithms to synchronize their internal states and share data structures to ensure the scalability of control planes, as well as to provide applications with a centralized view of the network state.

The controllers get closer to the switches as the number of controllers increases and the placement becomes more distributed, and the volume of CS traffic decreases. Conversely, if the controllers are fewer and more concentrated, the volume of CC traffic decreases.

Controller Placement Problem (CPP) is similar to a facility location problem and is also known as location analysis. This technique has been recently addressed by splitting the network into many internally connected subnetworks using a clustering mechanism. These approaches facilitate finding the optimum global solutions. Several researchers have used a heuristic approach to solving the CPP. Since the heuristics method is not a simple strategy for finding local optimal problem solutions and it is time-consuming. For this reason, clustering-based controller placement is more reliable for achieving valuable information. The distributed controllers' placement that is based on clustering algorithms performs better than the non-clustering method of placement [7]. Other studies have found that a clustered has better performance than a non-clustered network, providing less average latency and packet loss [3].

Several CPP formulations take into account a reliability value assigned to each switch, link, and controller, and aim to improve overall control plane reliability by optimally placing the controllers. Because network failures can interrupt communication between network

components (such as controllers or switches), it is critical to consider the reliability of networks when deploying controllers, as this can result in severe packet loss and performance degradation.

1.2 Thesis Overview and Contributions

The creation of SDN provided the centralization of the control plane, global network view, ease of complex configuration and network programmability, and the possibility to design new services that facilitate magnification of legacy network performance, circumventing former limitations and constraints. Multiple research topics, such as switch and controller design, scalability, latency, reliability and resilience, are still under study to allow their maturation and full SDN network deployment. The work presented in this dissertation has been developed under the framework of placing controllers in the Software Defined Networking (SDN). This thesis focuses specifically on applying in two network topologies: *Internet*2 *OS*3*E topology* and *Western European NRENs Topology*.

The section contains a summary, as well as a brief discussion of the scope and main contributions of this thesis.

Chapter 1: Introduction

The first chapter consists of an introduction to the full thesis. It clearly cites the statement of the problem, the objective of the study, the related work, the scope, and the methods that this thesis will use to achieve its objective, as described in summary below.

Chapter 2: Introduction to Software Defined Networks (SDN)

In the second chapter, a general overview of SDN is presented, and the role of SDN in future and 5G networks, as well as its main characteristics, and SDN Network Architecture and Components. There is also a discussion of the relationship between Network Function Virtualization (NFV) and SDN.

Chapter 3: Control Plane Optimization in SDN: State of the Art

The third chapter describes in detail the parameters used for optimal controller placement and optimization models.

Chapter 4: Multi-level Hierarchical Controller Placement

This chapter includes an extensive evaluation of state of the art solutions for the controller placement problem (CPP) [6]. This study enlarges previous methodologies using K-means and K-center algorithms to solve the CPP with a Multi-level Hierarchical Controller Placement Problem (HCPP), whereby the Super Controller (SC) is on top, some Master Controllers (MCs) are in the middle, and Domain Controllers (DCs) are at the bottom. Furthermore, this work includes results for a modified version of the optimization problem with an ad-

ditional constraint, which is particularly valuable as a benchmark for the deployed HCPP, estimating the number of controllers required. These optimization problems are NP-hard and the problem is formulated as a Mixed Integer Linear Programming (MILP) problem. The optimization metric is the latency between CS as well as CC communications.

The performance is evaluated using real-world large topology Western European Networks (NRENs). Up to our knowledge, this is the first type of CPP study recently conducted by the National Research and Education Network (NREN) in Europe. The design methodology is executed from the bottom to the top, acknowledging administrative boundaries (each country has its own management policy, which is the major constraint of proposed solutions). For this reason, using a country-based approach (as per nationally defined boundaries), the real network topology is separated into clusters following political-administrative boundaries.

Chapter 5: Joint Latency Controller Placement of Control Plane

In this chapter, Joint Latency Controller Placement (LCP) is defined as an accumulated latency to solve CPP and CS and CC latencies, taken into account simultaneously. In other words, accumulated latency is the weighted sum of CS and CC latency. The proposed optimization problem is formulated as a MILP model under the constraint of latency. Our method's performance is evaluated using the Internet2 OS3E real-world network topology. The LCP model extends upon the original CPP model [6], with some additional constraints and terms for the objective function.

The objective is to minimize the overall accumulated latency and minimize the number of network controllers while optimizing their placement to achieve an optimal balance at the same time. We propose that the value of λ is selected according to the application of the control plane. Running different control plane applications could require different λ values. We provide a solution to determine the optimal placement of the controllers for each case based on the value of λ.

Chapter 6: Joint Latency and Reliability-Aware Controller Placement

In this chapter, Joint Latency and Reliability-aware Controller Placement (LRCP) is proposed as a method of simultaneously considering communication latency and communication reliability. The controller placement optimization model takes into account both latencies between switches to their controllers and the latency between controllers at the same time. This is referred to as joint latency. Furthermore, we evaluate the reliability impact of a single-link failure (SLF). This is referred to as reliability-aware. We formulate the LRCP, which is more inclusive and easily adjustable than existing research work, including with regard to single-link failure (SLF). It is mainly geared toward reliability when a link failure event occurs in the control plane. It addresses the original CPP by extending it and taking communication latency and communication reliability into account.

Experiments were conducted utilizing Internet2 OS3E network topology in order to compare our findings to earlier research. The LRCP metric is constructed as an extension of the CPP [6] and LCP [8] models, adding the reliability of single-line failures in response to research in the LARC model [9]. LRCP is defined to reduce the accumulated latency by integrating the two sub-objectives of reliability and joint latency. The optimization of LRCP is formulated as a MILP under both latency and reliability constraints. LRCP provides optimal placements while taking into account a kind of balance between CS and CC latencies at the same time, as well as a reliability tradeoff with preventive placements in the event of a single-link failure (SLF). Parameters λ and θ are used, respectively, for each goal and play a key role in the deployment of the controllers. Preventive placements consider the corresponding backup paths.

Chapter 7: Evaluation of Joint Controller Placement for Latency and Reliability-Aware Control Plane

This chapter includes an evaluation of the optimization model of the LRCP. The evaluation metric is Control Plane Latency (CPL), which is defined as the sum of average switch-to-controller latency (CS) and average inter-controller latency (CC). The control plane latency is computed for each optimal network placement using the real latencies of the real network topology. The major contribution is the evaluation and analysis of the goodness of LRCP preventive placements in a real deployment. The CPL metric is used to quantify the assessment.

Chapter 8: Conclusions and Future Work

This chapter concludes the thesis, summarizing the most salient findings from the research and identifying areas for future investigations.

In addition, Appendix A contains a complete list of publications derived from the research conducted during the development of this thesis.

Chapter 2

Introduction to Software Defined Networks (SDN)

2.1 SDN Role in Future and 5G Networks

Fifth Generation (5G) networks are still under investigation and need more research in various dimensions. Since 5G technology is still in its early stages of development, experts are actively investigating various architectural paths to solve its main drivers. One of the principal building blocks and main challenges for 5G networks is the design of flexible network architectures, which can be realized by the SDN and NFV paradigms. SDN approaches have been identified as promising enablers for this vision of carrier networks, and will probably play a key role in the development of 5G networks. To meet the numerous challenges of future SDN-enabled 5G technology, a critical understanding of SDN is required.

In particular, SDN provides network administrators with the ability to manage network services through the abstraction of lower-level functionality [10]. This is accomplished by splitting the system into two portions. One portion decides where traffic is sent (the control plane). The other portion forwards data traffic to a selected destination (the data plane). Nevertheless, SDN is a logically centralized technology, and it is fundamentally an approach to networking in which the control plane is separated from the data plane. NFV, a complementary concept to SDN, allows the virtualization of entire network functions that are optimizing the network services themselves [11].

Mobile Generation Networks have progressed from the first generation (1G) to the second generation (2G) and third-generation (3G) to the fourth generation (4G) or Long-Term Evolution-Advanced (LTE-A) of mobile/cellular communications, with each generation, typically improving service and cost-efficiency [12], [13], [14]. 1G was deployed in the

early 1980s, with a data rate of up to 2.4 kbps. Nordic Mobile Telephone (NMT), Advanced Mobile Phone System (AMPS), and Total Access Communication System (TACS) were the main subscribers [15]. It has many drawbacks, such as low capacity, careless handoff, poor voice associations, and no security, because voice calls are stored and played in radio towers, increasing the vulnerability of these calls to unwanted eavesdropping by third parties [16]. These technologies were predicated on analogue systems, circuit switching, and Frequency Division Multiple Access (FDMA) radio systems [17].

In the late 1990s, the second wave was launched. Digital technologies are used for 2G mobile telephones. Global Systems for Mobile communications (GSM) was the first second-generation technology, mostly used for voice communication, with a data rate of up to 64 kbps [18], [19]. The battery of the 2G mobile phone (handset) lasted longer because of low-power radio signals. It also provides services (i.e., Short Message Service (SMS) and e-mail). The main benefits of technology were GSM, Code Multiple Access Division (CDMA), and IS-95 [20].

The 2.5G system uses 2G system frameworks (i.e., GSM and Enhanced DataRate for GSM Evolution, General Packet Radio Service (GPRS)) that were designed for packet switching along with circuit-switched voice applications. According to [21], the major technologies in 2.5G were GPRS, Enhanced Data Rate for GSM Evolution (EDGE), and Code Division Multiple Access (CDMA) 2000.

3G was established at the end of 2000. It has a transmission rate of up to 2 Mbps. 3G networks combine high-speed mobile access and Internet Protocol-based applications (IP). The only drawback of 3G phones is that they need more power than most 2G models. Along with this, 3G network services are even more expensive than 2G, since 3G involves the implementation and use of Universal Mobile Telecommunications Systems (UMTS), Wideband Code Division Multiple Access (WCDMA), and Code Division Multiple Access (CDMA) 2000 technologies. The evolution of technology such as High-Speed Uplink/Downlink Packet Access (HSUPA/HSDPA) and Evolution-Data Optimized (EVDO) has created an intermediate wireless generation from 3G to 4G called 3.5G and an enhanced 5-30 Mbps data rate. The combination of HSDPA and HSUPA is known as HSPA. 3G has developed through several updates, with the most recent release, HSPA+, being referred to as 3.75G.

The 5G networks are the newest generation of mobile technologies, designed to handle 100 billion connected devices, to be 10 to 100 times faster than the 4G mobile communications standard (LTE), and to support 1000 times more mobile data volume per geographical area [22]. The previous generation of cellular networks was quite different compared to 5G cellular networks, and the latter are required to support a multitude of devices and applications (i.e.,

autonomous vehicles, smartwatches, tactile internet [23] and Internet of Things (IoT)) [24], [25], [26].

According to the International Telecommunication Union (ITU), there are three kinds of service scenarios in 5G: enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communication (URLLC), and Massive Machine-Type Communication (mMTC). Different kinds of devices and application scenarios need more sophisticated networks, implementing low latency in data delivery, supporting high throughput, cost-efficient, and energy consumption, and having high scalability to handle a large number of devices, with ubiquitous and efficient connectivity for users.

In new generations of wireless communication networks, the most important assessment factor refers to the data rate metric. 5G will support eMBB with end-user data rates of 100 Mbps. However, the cell edge rate is about 1 Mbps in 4G, and now has to move to 100 Mbps in 5G. The peak data rate for 5G is designed around 10 Gbps, which is a 100-fold advance over 4G networks [27]. The latency of the 4G network is around 15 milliseconds (ms). The envisioned end-to-end latency for 5G is around 1 ms [28]. Another important goal of 5G networks is to reduce energy consumption 1000 times per bit [29].

Network scalability has become a significant factor in designing and developing the next generation wireless communications network to support the expanding quantity of mobile devices that connect to the wireless network and communicate with each other. The 5G network should be scalable with higher capacity than previous generations. By 2020, the number of devices connected to cellular networks is expected to be 50 billion. It is expected that all the connections implement security standards on authentication, authorization, and accounting. Due to heavy traffic, 5G is intended to improve reliable service and connectivity in crowded areas such as metro stations. Therefore, in 5G, an intelligent handover is needed with the slightest delay while switching the networks. The 5G idea is being explored around the world.

Low latency and high reliability are critical in several real-time applications, e.g., message transmission by robots monitoring patients, life safety systems, cloud-based gaming, nuclear reactors, sensors, drones, and connected transportation systems. However, it is a challenging task to achieve ultra-low latency and reliable data transmission (high reliability) on a wide-scale network without increasing network service costs, as it causes the creation of techniques that allow fast connections, quick handovers, and high data transfer rates.

In the forming of the 5G network, several factors may play an important role [30], [31]. From the technologies shown in Figure 2.1, the 5G architecture solutions are heavily reliant on two: Software-Defined Networking (SDN) and Network Function Virtualization (NFV).

Fig. 2.1 Multiple Technologies for 5G Networks

NFV and SDN concepts and architectures permit operators to decrease network costs, simplify the deployment of new services, diminish deployment time and scale their networks [32]. Details of SDN and NFV are described in the following sections.

1. Wireless Software Defined Networking (WSDN)
2. Network Function Virtualization (NFV)
3. Millimetre Wave technology
4. Massive MIMO
5. Network Device Density
6. Mobile Cloud Computing (MCC)
7. Internet of Things support

8. Device-to-Device (D2D) communication
9. Green Communications
10. New Radio Access Techniques

2.2 Main Characteristics of Software Defined Networking (SDN)

The conventional distributed network architecture has found it difficult to add new functionality due to the tight coupling between the control and data planes, and the whole structure is highly decentralized. Traditional networks are complex and difficult to administer. For data networks and the next-generation Internet, SDN has been implemented. It has been identified in several ways. The clearest and most established definition is provided by the Open Networking Foundation (ONF). SDN is one of the most important architectures for managing large-scale networks [33]. SDN has been applied as a new approach to designing, building and managing networks that separates the network control and forwarding planes and allows better optimization of each [34]. The main purposes of the SDN are to:

(i) Decouple the control and data planes of the network devices, like switches and routers.

(ii) Enable the control plane to be programmed directly from an open interface, such as OpenFlow.

(iii) Propose novel network control functionalities depending on an abstract representation of the network.

There are several reasons for using SDN:

- Virtualization: Using the network resources by ignoring physical location, how much it is, how they are managed, etc.
- Orchestration: By using one command, thousands of devices can be managed and controlled.
- Programmable: Network control is directly programmable since it separates the control plane from the forwarding plane (data plane). SDN provides the control plane to be programmed using a variety of software development tools along with the function of customization of the control network based on user requirements.

- Dynamic scaling: Ability to change in size and quantity.
- Automation: Lower OpEx such as troubleshooting, reducing downtime, policy enforcement, provisioning/re-provisioning/segmentation of resources and adding new workloads, sites, devices, and resources.
- Visibility: Monitor resources and connectivity.
- Performance: Optimize network device utilization (i.e., traffic engineering/ bandwidth management, capacity optimization, load balancing, high utilization, and fast failure handling).
- Multi-tenancy: This needs the complete monitoring and control of several parameters (i.e., addresses, topology, routing, and security). Essentially, it is sharing expensive infrastructure.
- Service integration: Load balances, firewalls, Intrusion Detection Systems (IDS).
- Openness: Full choice of modular plug-ins.
- Unified management of computing, networking, and storage.

2.3 SDN Network Architecture and Components

The fundamental planes of SDN are the data, control, and application planes. The application plane, which is connected to the control plane through northbound interfaces, defines network policies. The control plane uses southbound interfaces (i.e., Openflow) to communicate with the data plane. The data plane is liable for managing the forwarding data packets (i.e., switch or router). According to the Open Networking Foundation (ONF) [35], the SDN Architecture includes three principal planes, as illustrated in Figure 2.2 and Figure 2.3, and described below.

- **Management (application) plane**: Comprises networking applications such as routing, monitoring, firewalls and load balancing, and defines rules and policies for the management plan.
- **Control Plane**: The logic whereby controllers execute forwarding behavior and handle traffic. Examples of the control functions include routing protocols, the configuration for network middle-boxes, such as firewalls configuration or load balancers configuration etc. Moreover, it is responsible for programming and managing the forwarding plane.

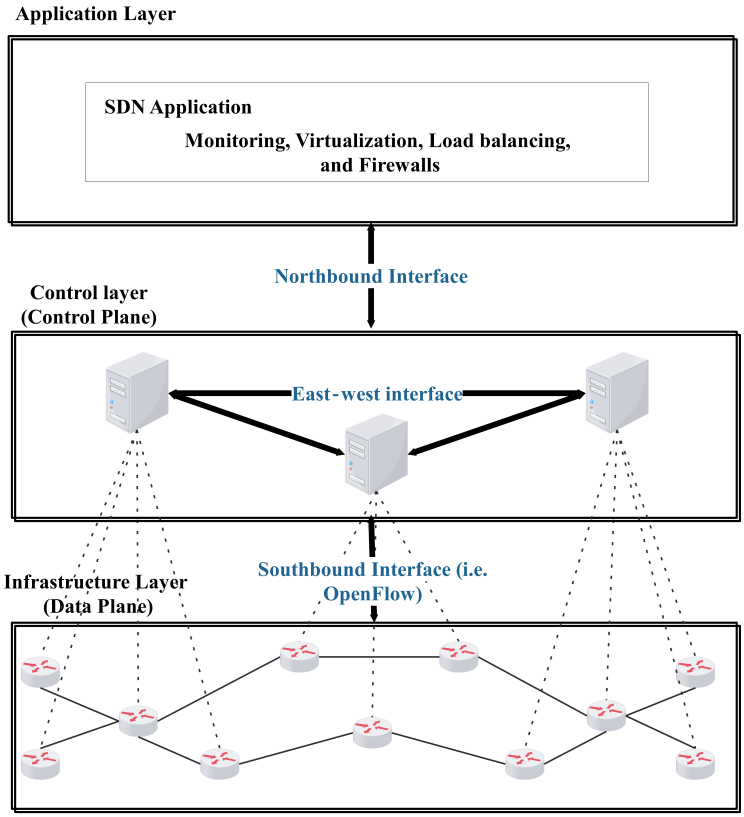

Fig. 2.2 Simplified view of SDN architecture

- **Data Plane**: Contains a set of one or more network components, each of which includes a set of traffic forwarding or traffic processing resources. It forwards the traffic based on control plane logic. An example data plane functions include forwarding packets, switching etc.
- **Northbound Interface (NBI)**: Programming interface that allows applications and orchestration systems to program network. The northbound interfaces (NBIs) are interfaces between the controller and applications or higher level layer control programs.
- **The East-West Interface Protocol**: Used for managing multiple controller communication, also known as a controller-to-controller interface (CCI). Communicating between controllers is the most challenging task. Recently, Tata Consultancy Services (TCS) presented research on inter-controller communication and published it as a white paper [36].

- **Southbound Interface (SBI)**: The Open Networking Foundation (ONF) has systematized the most well-known southbound interface, OpenFlow. Its principal function is to enable communication between the control plane and the data plane.

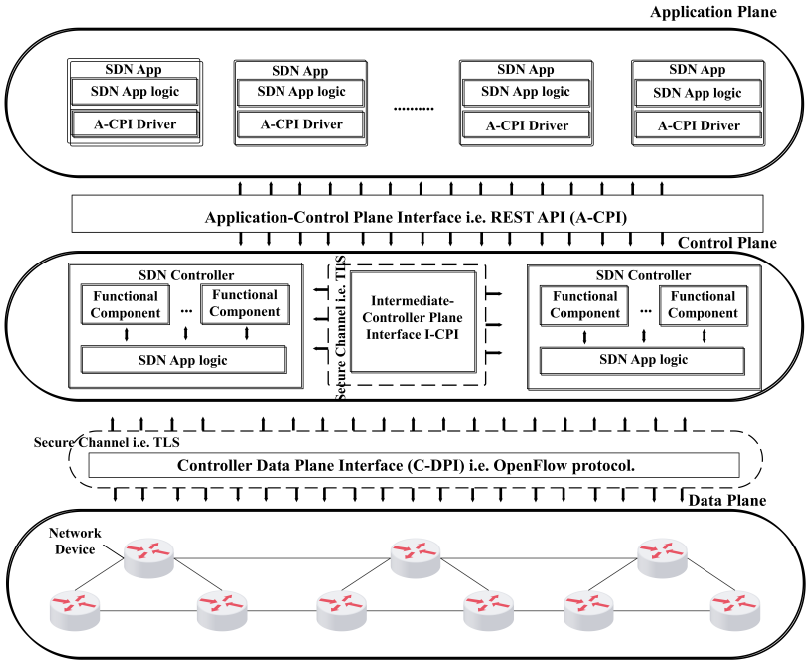

Fig. 2.3 Detailed view of SDN architecture

The data plane is the bottom plane, consisting of network devices (i.e., routers, access points, physical/virtual switches etc.) that are accessible and administered through C-DPIs by controllers. The network components and the controllers may communicate through secure connections, for instance TLS connections. The OpenFlow protocol is a standard C-DPI used for communication between the controllers and data plane devices.

The control plane is to supervise the network forwarding behavior through C-DPI. A set of software-based SDN controllers with control functionality has been developed. A controller comprises functional components and control logic. It has interfaces to allow communication through controllers in a control plane (Intermediate-Controller Plane Interface, I-CPI); using a secure channel (i.e., TLS) between network devices and controllers (C-DPI); and via the interface between controllers and applications (Application-Controller Plane Interface, A-CPI).

The application plane consists of one or more end-user applications (security and visualization etc.) that interact with controllers applying a simple view of an internal decision making process. Communication with controllers is mediated using open A-CPI REST API applications. An SDN application comprises an SDN App Logic and an A-CPI Driver.

SDN was popularized by the Opendaylight project, established in February 2013, directed by the Linux Foundation. This project is a highly accessible, modular, extensible, scalable and multi-protocol controller infrastructure assembled for SDN deployments on modern heterogeneous multi-vendor networks. Opendaylight affords a multi-driven abstraction service platform that allows users to write apps that simply operate across a wide variety of hardware and southbound protocols such as OpenFlow. When the Opendaylight project joined with open and published application programming interfaces (APIs), Opendaylight SDN Controller Platform (OSCP) offered the most flexible platform to expand universal, network-wide applications. OpenFlow (Figure 2.4) is the most usable SDN protocol/standard. This protocol is applied for managing the southbound interface of the SDN architecture. OpenFlow operates as a protocol to send/receive forwarding rules from controller to switches. There are three main key ideas of OpenFlow, which follow the characteristics of SDN:

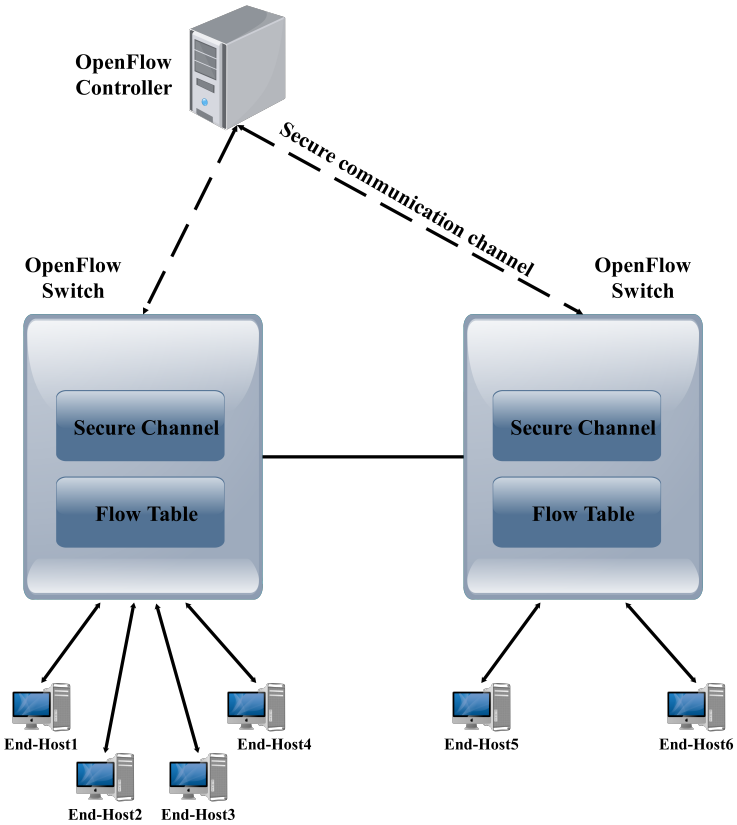

Fig. 2.4 OpenFlow model

(i) Separation of control and data planes, which is based on the concept of OpenFlow controller and OpenFlow switches.

(ii) Centralization of control, in terms of the OpenFlow controller.

(iii) Flow based control by maintaining a flow table based on traffic decisions.

(iv) Reducing OPEX and CAPEX.

The OpenFlow specification defines an open protocol for allowing software applications to program the flow tables of various switches. As shown in Figure 2.4, an OpenFlow architecture consists of three main components: an OpenFlow-compliant switch, a secure channel, and a controller. OpenFlow protocol [37] separates the controller and the forwarding devices. OpenFlow consists of two main entities: the OpenFlow switch (forwarding plane) and the OpenFlow controller (control plane). OpenFlow standardizes information exchange between the two planes.

The OpenFlow Switch contains one or more flow tables. A flow table is a set of flow entries for processing packets. A flow entry is utilized to match and process packets. Each flow entry consists of:

- Many matching rules or matching fields to matching packets. Match fields may have information to establish in the packet header, ingress port, and metadata value.
- Counters used to assemble statistics for the special flow, for instance the number of received packets, number of bytes and duration of the flow.
- A collection of instructions, or actions, to apply to matching packets, commanding how to handle matching packets.

OpenFlow switch utilizes an OpenFlow channel, which has three kinds of messages. The controller or switch message is sent by the controller and may not require a reply from the switch. The asynchronous messages notify the controller of a packet arrival, a change in the system's switch state, or a failure. Symmetric messages can be sent from both the controller and the switch for other purposes. The OpenFlow controller manages flow tables within the switch by adding, updating, and deleting flow entries.

An essential aspect of SDN is the connection between the data plane and the control plane. As forwarding components are controlled by an open interface, it is significant that this link remains available and secure. Southbound interfaces and northbound interfaces are SDN components with various liabilities. The OpenFlow protocol is the most extensively accepted and implemented southbound API for SDN-enabled networks, as it defines the communication between the control plane and network devices. A southbound interface (SBI) is an alternative OpenFlow protocol specification that allows a network component to communicate with a lower level component interface layer in an SDN.

The most important goal of a southbound interface is to implement communication and management among the network's SDN controller, links, physical/virtual switches and routers. It permits the controller to design the network topology, define network flow and perform several requests relayed from northbound application programming interfaces (APIs). The northbound interfaces are used to communicate between Controller and the services and applications running across the network, supplying higher-level abstractions to program different network level services and applications. In SDN architecture, the northbound interfaces are mainly a set of open source application programming interfaces (APIs).

SDN architecture makes it easier to manage flows in a high-quality system (compared to traditional networks). In a conventional network, flows (or packets) are essentially managed based on a single or a few attribute combinations of packet headers (i.e., longest destination IP prefixes, destination MAC addresses, or a combination of IP addresses and TCP/UDP port numbers etc.). SDN enables managing flows based on more attributes of packet headers by means of a Controller-Data Plane Interface (C-DPI), such as the OpenFlow protocol.

2.4 Network Function Virtualization (NFV)

NFV has drawn significant attention from both industry and academia as a significant revolution in telecommunications service provisioning. NFV has been designed as an approach to address by leveraging virtualization technology to present an alternative method to design, deploy and manage networking services. The major idea of the NFV is the separation of the physical network equipment from the function that runs on it (i.e., domain name service (DNS), firewalling, network address translation (NAT) and caching) [38]. There are several advantages of virtualization, such as scalability, Opex and Capex reduction, mobility, and quick provisioning.

Virtual switching will be a key function for many NFV deployments. Open vSwitch is a production-quality, multilayer open source virtual switch designed to enable effective network automation through programmatic extension, while still supporting standard management interfaces and protocols such as NetFlow, sFlow, SPAN, RSPAN, CLI, LACP and 802.1ag. The principal goal of Open vSwitch is to provide a switching stack for hardware virtualization environments, while supporting multiple protocols and standards used in computer networks. The NFV approach has several important requirements in the following areas:

- General: Partial or full virtualization.
- Portability: Separated from underlying infrastructure.

- Performance: Performing the capabilities required to characterize the infrastructure requirements for particular performance base of software functions and facilities to monitor and predictable performance.
- Elasticity: Scalable to meet service level agreement (SLA), and movable to other servers.
- Resiliency: Able to recover after failure. Specified packet loss rate, call drops, and time to recover etc.
- Security: Role-based authorization, authentication.
- Service continuity: Seamless or non-seamless continuity after failures or migration.
- Service assurance: Timestamp and forward copies of packets for fault detection.
- Energy efficiency requirements: Addressing the technical capabilities that will support decreasing the energy consumption of large scale virtualized networks.
- Transition: Coexistence with legacy and interoperability among multi-vendor implementations.
- Service models: Operators may use NFV infrastructure operated by other operators.

As shown in Figure 2.5, NFV architecture according to the European Telecommunications Standards Institute (ETSI) consists of four key elements: NFV infrastructure (NFVI), virtual network functions (VNFs), hypervisors, and NFV management and orchestration (NFV MANO) [39], [40], [41]. The principle element here is the VNF, which is a software accomplishment of network functions, running on a generic cloud infrastructure. VNFs have grown as a result of the NFVI, which includes virtual computation, virtual storage, and virtual network resources. Virtual resources are constructed using hypervisors to virtualize over physical hardware resources within the network. Hardware resources could include networking (switches and Remote Radio Head (RRHs)), storage, and computing resources.

The NFV MANO framework controls the supply of VNFs, the composition of VNFs, and the infrastructure they run on. MANO can also chain several VNFs to validate an end-to-end service. NFV has sufficient power for network appliance multi-version and multi-tenancy, while employing a single platform for various applications, users, and tenants, enabling a wide variety of eco-systems with openness. On the other hand, the NFV improvement for wireless systems is still under research and development. NFV has become a complementary approach to SDN for network management [42]. However, both approaches depend on

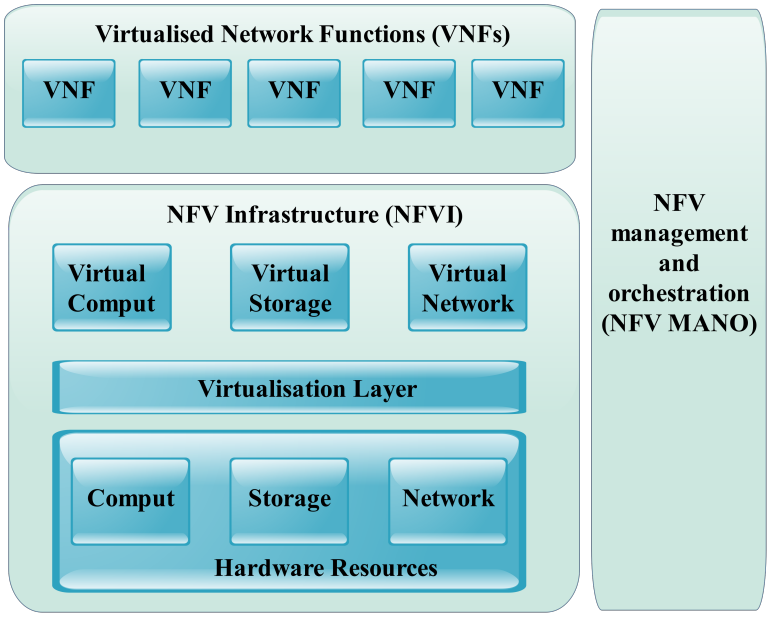

Fig. 2.5 NFV architecture

various methods for the same target that is managing networks. The relationship between NFV and SDN is as follows:

- SDN decouples the control and forwarding planes with a centralized view of the network, and NFV optimizes network services themselves.
- The purpose of SDN is decoupling the handling of packets and connections from whole network control. NFV separates NFs from the particular hardware component.
- In SDN architecture, virtualization is the allocation of abstract resources to specialized customers or applications. NFV seeks to abstract NFs aside from devoted hardware.

Chapter 3

Control Plane Optimization in SDN: State of the Art

3.1 Taxonomy of Control Plane Approaches in SDN

By decoupling the control plane and data plane, the SDN approach can simplify network management and speed up network innovations [43]. The major benefits of SDN are its programmability and agility. Scalability issues in the control plane (controller) are one of the key problems that require more research attention [44], [45], [46]. In the meaning of SDN, scalability is defined by the two pre-eminent metrics: throughput and flow setup latency. In an SDN context, the number of flow requests handled per second is referred to as throughput, while flow setup latency refers to the time it takes (i.e., latency) to respond to flow requests. The principal reasons that make the control plane a scalability bottleneck in SDN are decoupling of the control plane from the data plane, quantity of events/requests handled by a controller, and controller-switch communication delay. The proposed control plane scalability approaches [35] are categorized into two categories with sub-categories, as presented in Figure 3.1.

1. **Topology-related approaches**: Topology-related approaches are the relation between the topology of architectures and scalability issues, with sub-categories of centralized (single) controller designs and distributed approaches. Distributed approaches have the subcategories of distributed (flat) controller designs, hierarchical controller designs, and hybrid designs. These categories are summarized below:

   (a) Centralized (single) controller designs

   (b) Distributed approaches

i. Distributed (flat) controller designs
ii. Hierarchical controller designs
iii. Hybrid designs

2. **Mechanisms-related approaches**: Mechanism-related approaches propose several ways of optimization for controllers and application implementations, with subcategories of parallelism-based and control plane routing scheme-based optimization:

   (a) Parallelism-based optimization

   (b) Control Plane Routing Scheme-based optimization.

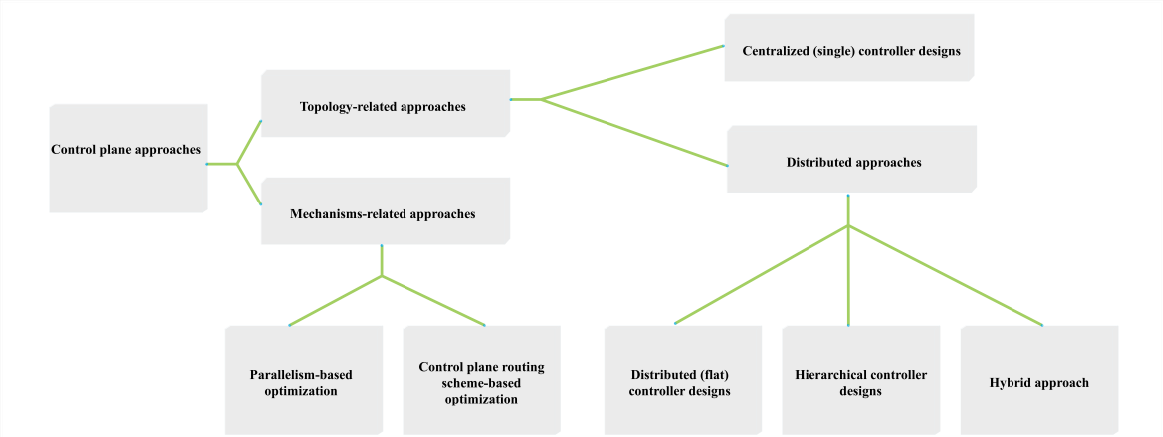

Fig. 3.1 Taxonomy of control plane approaches in SDN

An SDN involves centralized control over distributed network architectures [47]. In topology-related architectures, it is very important to describe the two-way data path between network devices, the control path among controller and data devices, and the controller-to-controller path among controllers.

- In Figure 3.2 (centralized (single) controller design), there is one major controller with a global network state.
- In Figure 3.3 (distributed (flat) controller design), each controller is liable for various sites/parts of the network(s), with partial or full shared network view.
- In Figure 3.4 (hierarchical controller design), there are levels in which controllers are responsible for various sites (subnetwork/domain) of the whole network, and a root controller on top with a global network view for global applications such as routing.
- In Figure 3.5 (hybrid design), data plane devices are also involved in network control.

The principal network components comprise switches, controllers, and links that connect devices. With the popularization and deployment of SDN, the problem of controller placement is increasingly pertinent. Controller placement is a fundamental concern when designing a distributed SDN control plane and deciding on the number and placement of controllers [48], [49], [50], [51], [1]. A significant parameter to consider while doing so is the propagation delay between the controllers and the network devices, especially in the context of a large-scale network. A diversity of solutions has been introduced to tackle the controller placement problem in SDN. The objectives involve reducing the latency between controllers and their associated switches, improving the reliability and resilience of the control network, and minimizing deployment cost and energy consumption.

3.1.1 Centralized (Single) Controller Designs

This type of centralized network architecture configuration is designed around a single central controller that handles all the major processing with a global network view [52]. This architecture's design is easy to develop, and the network is simple to administer.

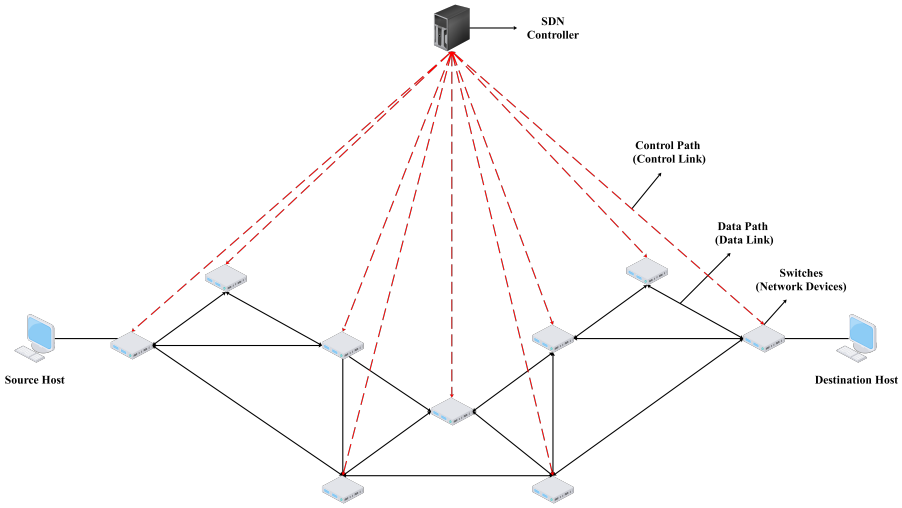

Fig. 3.2 Centralized (single) controller design

The design of this architecture may be suitable for small and medium-sized networks. However, centralized network architecture is not sufficient to manage the burden of environments such as data centers and large-sized networks because of the number of events/requests to be handled by the controller. Thus, in comparison with the distributed (flat) controller, hierarchical controller and/or hybrid designs, a single control architecture is less scalable due to high network traffic, which may easily over-load the controller.

3.1.2 Distributed (Flat) Controller Design

In this structure (Figure 3.3), the network is split into multiple domains (i.e., subnetworks), and each controller manages each domain within the entire network. For distributed controller architectures, there are two strategies to construct the controller's network view, in each of which the controller must communicate through controller-to-controller channels to exchange the necessary status information on their domains:

(i) **Local view approach**: Enables each controller to have its own local network view of the topology, allowing for local decision-making in the local view approach, and each neighboring local network can be abstracted as a logical node.

(ii) **Global view approach**: Each controller manages the global view of the entire network.

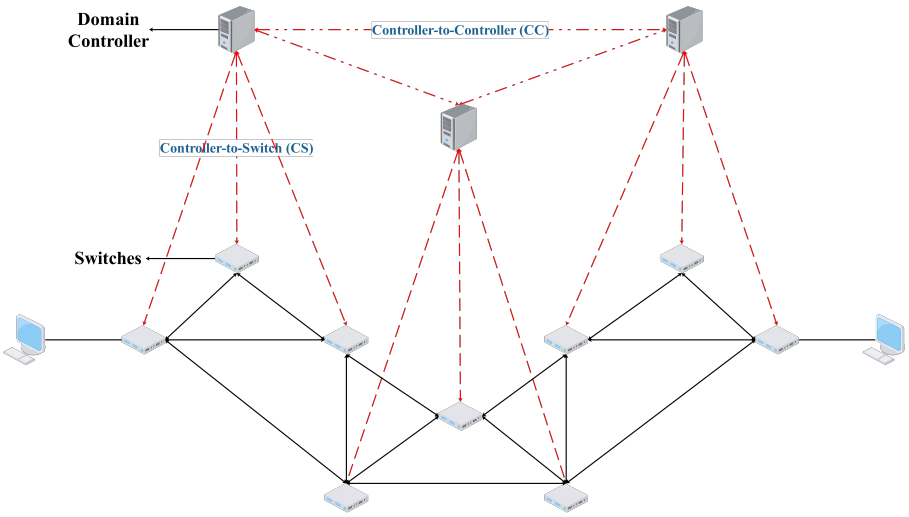

Fig. 3.3 Distributed (flat) controller design

3.1.3 Hierarchical Controller Design

Local controllers address local applications requirements in a hierarchical architecture, which does not require a global view of the network, while the top controller, also known as Root Controller, manages other applications requiring the global network view of the topology (Figure 3.4). This configuration differs from the flat architecture of controllers in the network view. In the hierarchical architecture, the controllers are at the lower levels with no network-wide view, while controllers have a network-wide view of the network in the flat architectures. The horizontal and vertical approaches are two methods for fully distributed design:

(i) **Flat approaches**: In a horizontal (flat) control plane, multiple controllers are organized with the network devices (i.e., switches) operated by each controller.

(ii) **Hierarchical approaches**: In the vertical architecture, the controllers are arranged vertically, and each manages a set of switches in the hierarchical data plane.

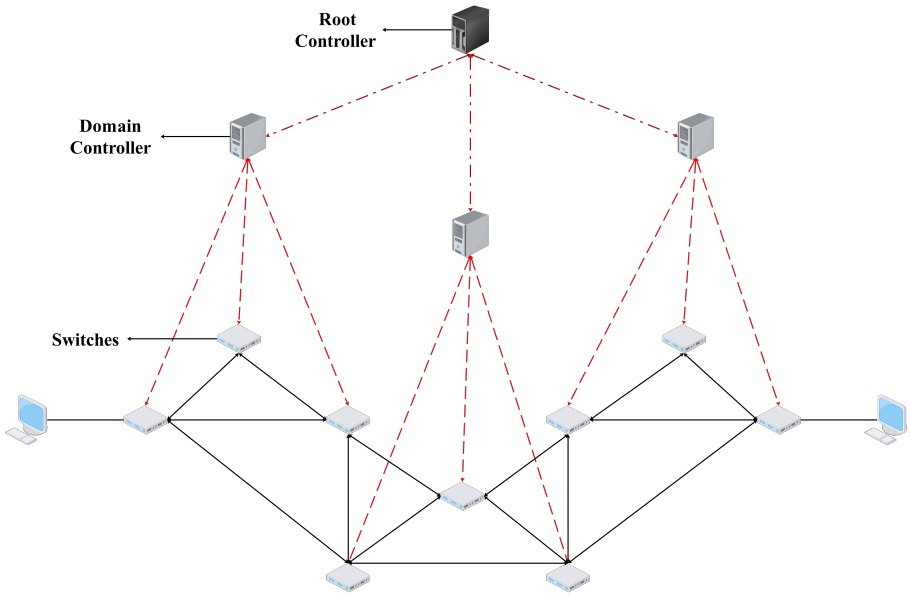

Fig. 3.4 Hierarchical controller design

The hierarchical control plane or fully distributed architectures of multiple controllers do not require the synchronization of controllers to each other. The root controller has a global view of the entire network in this architecture and carries out routing operations. The other controllers are responsible for managing their domains. Events such as inserting flows, and modifying and deleting in flow tables, can happen from a local view or global view. Events are identified as local events under a centered physical controller, and events within the distributed controller architecture are defined as global events, influencing each flow table to exist in all controllers. The nearest controller in the hierarchy handles local events, while global events are handled at a higher level in this deployment model.

The flat structure means that the network is horizontally partitioned into multiple domains, each of which is governed by a particular controller in charge of managing a subset of switches. The organization of controllers in such a flat architecture has several advantages, including reduced latency and improved resiliency. All controllers are at one level in the flat architecture, and communication between controllers is performed to gain a global view. The hierarchical control plane architecture implies that, according to the services required, the

control plane is vertically partitioned into multiple levels (or layers). A hierarchical control plane organization can enhance the scalability and performance of control plane. Controllers are constructed according to a tree structure in the hierarchical control plane architecture, so that not all controllers are at the same level.

(i) At the **lowest level**, the controllers directly manage the network devices associated with the controller, and contain only their local network view.

(ii) The **upper-level** controllers provide guidelines for actions and decision-making to manage the view of all the lower-level controllers directly.

(iii) The **root controller** has a global view, and is responsible for coordination between the local controllers.

3.1.4 Hybrid Design

A hybrid architecture is similar to the hierarchical architecture, except that multiple controllers at the highest level may communicate with each other as a distributed (flat) architecture, due to which this approach is termed "hybrid". The hybrid approach is designed to reduce controller overload (because the overloading of controllers causes processing problems and data delays). Controller overload can be mitigated by incorporating data plane devices in the control plane.

The hybrid hierarchical control plane [53] is a combination of distributed (flat) and hierarchical architectures. It seeks to combine the benefits of both designs, and to combine them into a hybrid structure in order to avoid the problems faced separately by each architecture. It focuses mainly on the path stretch problem and super linear computation complexity caused by these two architectures.

The difference between the best optimal path and the actual path traffic takes in the network is referred to as path stretch. This issue arises in hierarchical architecture. Superlinear computational complexity growth is caused by distributed (flat) architecture when the network scales to a large size.

The hybrid hierarchical architecture of Orion [54], which forms the conceptual basis for the hierarchal approach adopted in the current study, addresses these issues by dividing architecture into three layers: physical layer (bottom layer), area controller layer (middle layer), and domain controller (top layer).

The physical layer is the bottom layer, which consists of large amounts of connected OpenFlow switches. The area controller collects information from OpenFlow switches and links it to domain controllers, as well as managing the intra-area topology and processing

intra-area routing requests and updates. The top layer is the domain controller layer, which considers area controllers to be devices, reducing the path stretch problem and super linear computation complexity, synchronizing the global abstracted network view through a distributed protocol.

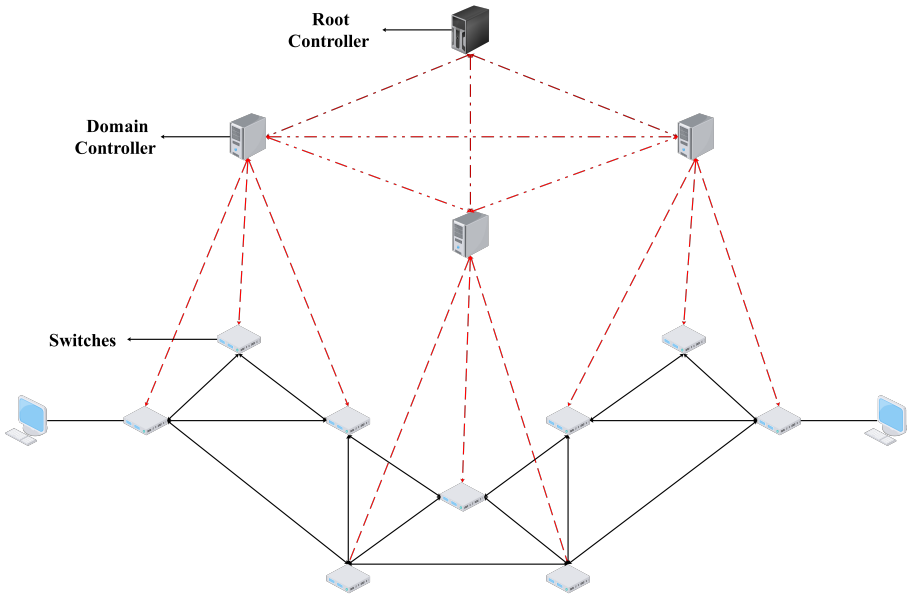

Fig. 3.5 Hybrid design

3.2 Control Plane with Distributed SDN Controllers

The control plane is a logical network that overlaps the physical network, consisting of controllers and switching devices that are placed at the same location. The control network can be split into multiple domains, each of which is managed by some controllers. As shown in Figure 3.6, there are two types of paths: one is a data path, including switch-switch connection (SS), and the other one is a control path, including switch-controller connection (CS) and inter-controller or controller-to-controller connection (CC). However, note that the control traffic path is transmitted and spread through data paths.

Two control planes can be defined for a network under the administration of distributed controllers.

(i) The **switch-to-controller communication (CS)** supports communication between any switch and its assigned controller through the controller's southbound interface. This interaction is typically dedicated to commands on the data plane (i.e., via the OpenFlow protocol), and to configuration and administration of network switches.

(ii) The **controller to-controller communication (CC)** allows direct communication among the controllers through the East-West interface. The distributed controllers also need to synchronize the shared data structures to guarantee a consistent view of the global network.

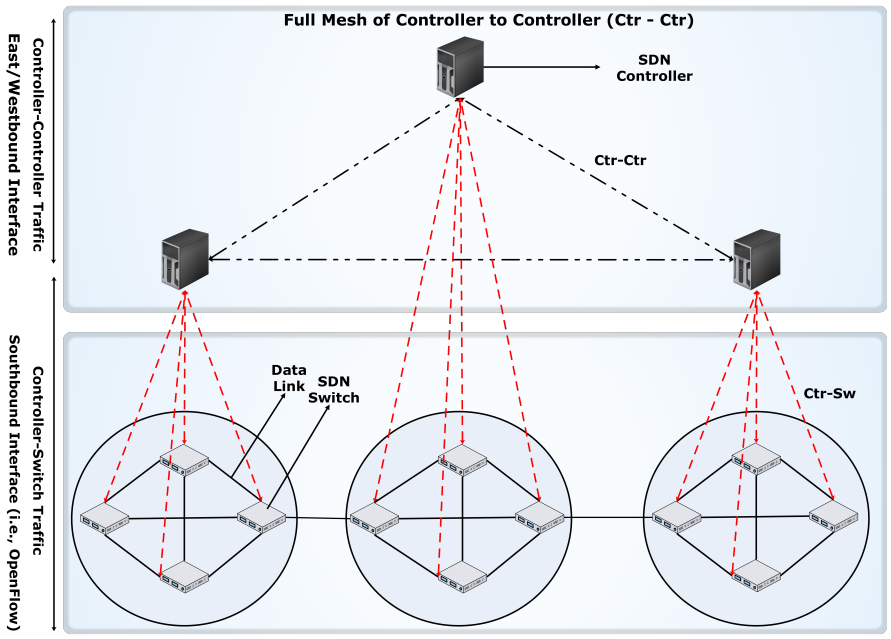

Fig. 3.6 SDN Control Plane Architecture

The response time of packets transferred from the controller to each associated switch or switch to controller is usually shorter than the response time for the traffic communication between controller and controller (CC). CC traffic is generally very complex because distributed controllers adopt coordinated protocols and algorithms to synchronize shared data structures in order to ensure a consistent view of the global network, and to allow a central view of the network state of applications. The controller-switch traffic in the control plane depends mainly on the network application running on the controller. Communication via the southbound interface (CS traffic) is faster than communication via the east-west interface (CC traffic). Depending on the control plane application, minimizing CC latency may be more important than CS traffic communication.

3.2.1 Switch-to-Controller Latency (CS)

Since the network's control logic is decoupled from simplified switches, and all network functions are carried out through message exchange between controllers and all assigned switches, latency is particularly critical. The distributed controller in the control plane is the key component used for all operations of data plane management [55]. A controller has a global view of the entire network, including devices from a data plane. It connects these resources with management applications and conducts flow actions among the devices based on application policy. The latency value reflects the distance between the switch and the controller for packet transmission latency. When the switch sends the packet transmission latency to the controller, the switch can be allocated to the controller and be used to update the switch configuration of network events and changes to the controller.

3.2.2 Inter-Controller Latency (CC)

In the conceptual centralized controller, the distribution of controller implementation comprises multiple controllers, and the controllers are connected [56]. As a logical centralized system, the controllers enable the exchange of required information for synchronization among the multiple controllers. The controller possesses a global view of the entire network topology, and provides centralized network control and management [57], [58], [59]. A specific controller manages each switch. When a controller wishes to transmit messages to a switch controlled by another controller, the controllers must communicate among themselves. Hence, inter-controller communication influences the network performance of end-to-end communications between disparate switches managed by different controllers. Therefore, reducing inter-controller's latencies (CC) is necessary to reduce switch-to-controller latencies (CS) [60]. However, for topological reasons, minimizing one type of latency entails maximizing the other, and vice versa. We focus our investigation on the latency tradeoff achievable in the CS and in the CC control planes.

3.3 Controller Placement Problem (CPP)

A single controller manages and monitors the network efficiently and provides networking functionalities such as flow rules that are managed by the controller, routing decisions, policy-based network management on the global network view [61]. The main drawbacks of CPP include latency between switches placed away from their assigned controllers, power processing abilities, link cost in network design, and reliability issues. In this context, key challenges are handled by distributing multiple controllers across the network to define

a scalable control network. The communication cost among controllers is to manage a consistent global view of the network among multiple controllers and to guarantee proper network operation.

The controllers are installed at different locations in the network to provide optimal performance, efficient cost management and minimum cost. At any point in time, the switch set of any controller receives flow rules from the later. Determining controller locations and the optimal number of controllers, and assigning a set of switches to each controller in the network, is known as the CPP, which includes to controller placement (CP) and controller number (CN). CP indicates where controllers are placed in the control plane [6], while CN indicates the number of required controllers in the control plane. CP and CN are used in combination. The placement of controllers imposes several challenges in the control plane:

1. How many controllers are required for the entire management of the network?
2. How many switches can each controller manage?
3. What are the appropriate locations to place the controllers?
4. How can the switch be managed by another controller during the failure of controllers?

As shown in Tables 3.1 and 3.2, several CPP issues have been addressed regarding various constraints of the network, such as control latency, reliability, resilience, control load balancing, cost-efficiency, information sharing, and multiple objectives.

Heller et al. [6] established the first research in this field, launching the CPP concept for the first time. They identified that the number of controllers and their placement (required for a given topology) affect latency, and they attempted to reduce the average and maximum (worst case) switch-to-controller latencies with a structured CPP, using k-median and k-center algorithms. However, they did not take into account multiple controllers in the network. Singh et al. [2] presented a survey on CPP research work from 2012 to 2017, critically analyzing existing solutions and identifying limitations and future scope. Wang et al. [110] showed a critical analysis of state of the art controller placement solutions that analyzed various parameters, such as latency minimization, maximizing reliability and performance, deployment cost, and energy consumption

3.3.1 Solutions and Strategy

In SDN, the placement of controllers is a complex problem of optimization. In large-sized networks, there are several appropriate CPP approaches. Finding the optimal solutions is time-consuming, and different solution methodologies have been proposed Mixed Integer

Optimization	Paper	Methodology
Control latency	[57]	BF, EA
	[62]	MILP, GA
	MDCP [63]	IQP, GA
	CPP [6]	BF (BF)
	[64]	Independent Dominating Set
	MC-SCA [65]	QIP, ILP
	CNPA [66]	K-means
	OKMP [67]	K-means
	HPP, VCPP, JHCP [68]	ILP
	[69]	ILP
Reliability	MCC [70]	Min-cover, PSO
	[71]	ILP, GA
	RCP [72]	K-means, GA
Resilience	[73]	Min-cut
	CCP, FFCCP, CO-FFCCP [74]	ILP
	CNCP [75]	MILP
	[76]	ILP
	RCP-DCP, RCP-DCR [77]	MILP
	[78]	GA
	[79]	ILP
	[80]	SA
	GROM, LROM [81]	SA
	[82]	ILP
	CPCNS, CPSLF [83]	BF, GA
	CNM [84]	ILP
	LFACCP [85]	ILP
	RMM-FB, RMM-MC, RMM-MB, RMM-LM.[86]	ILP
Control load balancing	CPC [87]	Capacitated p-center problem
	CCPP [88]	Custom
	[89]	Non-zero sum game
	[90]	GC
	[91]	Custom
Cost-efficiency	RCCPP [92]	ILP
	HCS [93]	GC
Information sharing	LLDP [94]	GC
	[95]	GC

Table 3.1 Controller placement approaches

Linear Programming (MILP), heuristic, and clustering solutions can be implemented, as addressed below.

MILP Solutions (exact solutions)

This approach uses MILP tools to solve CPP and find optimal control placement solution for a given network. This linearization approach requires computing for each controller location, latency among the controllers, latency between the controller and the switch assigned by the

Optimization	Paper	Methodology
Multi-objective	[96]	Custom
	MLkP [97]	Multilevel k-way partition
	CLPA [98]	Label propagation algorithm
	[99]	k-center, k-medoids, greedy-SA, spectral clustering, PSO, FFA
	DCPP [100]	ILP, SA, Greedy algorithm
	[7]	DBCP
	HCPM, HRA, CLACPM, CPAPLS[101]	Jain's Fairness Index, Maximum SC/CC latencies
	MHNSGA [102]	Link load indicator, Maximum link utilization of control paths
	[103]	Robustness Metric, k-Critical and Fast Failover heuristics
	PAM [104]	Partitioning around medoids, NSGA-II
	MCPS [105]	ILP for MCPS
	MCDA [106]	Reference level decision algorithm
	MOCP, ABFO[107]	Adaptive Bacterial Foraging Optimization
	AHP, CPGA [108]	GA
	CPP PSO, CPP FFA[109]	k-center, k-medoids, greedy-SA, spectral clustering, PSO, FFA

Table 3.2 Multi-Objective Controller Placement

controller, and reliability and network resilience. This can be optimally solved by Gurobi [111] and CPLEX [112] solutions. However, the disadvantage of this strategy is that it is time-consuming to find the optimal solution in large-scale networks.

Clustering solutions

Clustering is a serviceable tool for identifying homogeneous groups of objects based on the values of their attributes and is utilized in several disciplines and applications. Clustering can be used with both heuristic and exact solutions, such as MILP to solve the CPP. The clustering approach reduces the search space and identifies the feasible controller placement set. An heuristic or exact solutions approach is used to choose the best placement of the controller. The clustering approach intends to improve the regional distribution of the switches, reduce computational complexity, and allow for enhanced load balancing among all clusters. Clustering is an approach which groups objects into appropriate and reasonable groups. According to a certain metric, objects within a particular cluster are more similar to each other than to objects in different clusters.

When implementing SDN in a real-world large network, it may be divided into many small network domains, each of which is controlled by a single controller. Network partitioning can reduce the management complexities of large-scale network partitions. This might apply to different network aspects, including scalability, privacy and deployment. As shown in Figure 3.7, the central idea is to partition the entire network into several domains and assign a controller to each domain [113]. A domain can be a sub-network in a data center, or network such as an Autonomous System (AS) or an enterprise network.

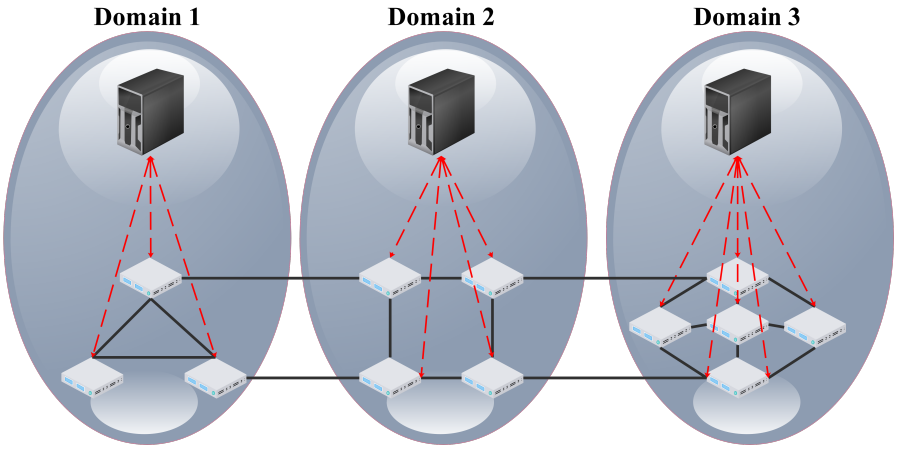

Fig. 3.7 Partitioning the entire network into multiple domains

Optimization	Paper	Methodology
Network partitioning	[114]	Min-max cut function(Mcut)
	[66]	K-means algorithm
	[115]	K-means algorithm with the cooperative game theory
	[7]	DBCP
	[116]	Pareto-based Optimal Controller Placement (POCO)

Table 3.3 Network partitioning based controller placement

3.3.2 Objective Functions

Reduce control plane latency

In an SDN-enabled network, the latency between controllers and switches is especially critical since the control logic of the network is separated from simplified switches, and it carries all the functions of the network out through message exchanges among controllers and switches. The network latency between controllers and switches plays a most significant role because it profoundly influences the overall performance of control plane [117]. The latency comprises the packet transmission latency, packet propagation latency, queuing latency and switch processing latency. The packet transmission latency covers the ratio between the packet size and the transmission rate of a link. The length (distance) between controllers and switches essentially determines the packet propagation latency. The switch queuing latency is presented by the congestion level of a link. Loads of controllers fundamentally influence the controller's processing latency. One challenge is to choose suitable locations for controllers to shorten the latency between controllers and associated switches. As shown in Table 3.4, following a thorough analysis, we categorized the latency into four aspects:

(1) the average switch-controller latency (SC-avg latency), (2) the worst switch-controller latency (SC-worst latency), (3) the average inter-controller latency (CC-avg latency), and (4) processing latency.

Optimization	Paper	Methodology
Average switch-controller latency	CPP [6]	K-means, k-center
	[69]	MILP
	POCO [118]	Pareto SA
	[7]	Density-based switch clustering algorithm
Worst switch-controller latency	OKMP [67]	Optimized k-means
	MOGA [119]	PSO
	[115]	Cooperative game
	[120]	FFA
Average inter-controller latency	[121]	Bargaining game
	[57]	ILP
	CLEP [122]	GA
	[123]	SA
Processing latency	CCPP [88]	Capacitated k-center
	[124]	Minimum spanning tree
	CNPA [66]	Improved k-means

Table 3.4 Latency aware controller placement based on [1]

Maximize Reliability

Reliability is attracting increasing research attention. Network failures could cause disconnections among controllers and the forwarding planes, and further disable some of the switches. It is of extreme significance to improve the reliability of control plane. Reliability is indicated as a performance metric that is inversely proportionate to the expected percentage of control path loss, and the optimization goal is to reduce the expected percentage of control path loss [125].

Using a multi-controller approach solves the single of point failure problem, but it cannot guarantee the high reliability of the control plane. The capacity of the connection links between switches and controllers is limited. In the event of congestion, interruption, or failure in these connections, controllers and switches normally cannot communicate with each other, resulting in the siloing of controllers and switches. Controllers could failed or be overwhelmed with malicious attacks (i.e., excessive packet-in requests). Therefore, the reliability of the multi-controllers is also important in the actual deployment. A complete control path comprises nodes (or switches), links (or edges), and controllers. Failures in the control plane influence network performance and to avoid failure, switches have to be connected to multiple controllers efficiently. Furthermore, the first approach is based on two

disjoint control paths, when node failure or link failure occur; and the second approach is to reduce the length of the control path, which includes fewer network components. .

Optimization	Paper	Methodology
Multiple control paths	[82]	ILP
	RCP, RCP-DCP, RCP-DCR[77]	MILP
	[72]	Clustering Algorithm
	[126]	Bully algorithm
Multiple controllers	[127]	Linear programming
	[128]	GA
	[75]	SA
	FTCP [129]	Heuristic algorithm
Minimize the control path length	RCP [130]	GA
	[131]	K-Critical
	[70]	Min-Cover
	RCCPP [132]	Clique-based approach

Table 3.5 Reliability aware controller placement based on [1]

1. **Multiple control paths**

   Control actions must be transmitted through the control paths (e.g., packet sending and flow entry distribution). Optimizing the control path is an efficient way to achieve control reliability in the control plane. In multiple control-path, there are at least two disjoint control paths between the switch and their assigned controller. When the main path fails, the system can switch to the backup path to ensure normal operations. Applying path redundancy can reduce the influence of single link or node failure by switching to an alternate path on the control path.

2. **Multiple controllers**

   In the event of a node failure, routes can be mapped or migrated to another node quickly, or flows can be rerouted on alternative paths that are not connected to the node. Unlike traditional network nodes, a controller handles traffic management in a network or domain and cannot be migrated or remapped. It would severely disrupt the operation of the network controlled by the controller if a controller failed or crashed. Therefore, studying the reliability of controller nodes has a significant impact on multi-controller reliability. In the control plane, a switch that connects to multiple controllers improves the reliability of the control plane by avoiding a single event of failure.

3. **Minimize the control path length**

   The control path comprises switches (nodes), links (edges) and controllers. Reducing the network elements included in the control path can improve system reliability.

3.4 General Formulation

The network is modelled as a graph $G = (V, E, L)$, where V represents the set of switches, E represents the set of links between switches and L represents the set of switch locations. To formulate the problem mathematically, L_{avg} denotes the average latency. Let $d(v,c)$ be the distance (or edge weights) between switches $v \in V$ and the controllers $c \in C$. The model optimizes the latency between switches and controllers. It is also important to note that the number of controllers must be less than or equal to the number of switches $n = |V|$. The controller is deployed at the location of a switch. The number of controllers is fixed to k where $k \leq n$. The set of controllers to be installed is represented as $C = \{c_1, c_2, c_3, ..., c_k\}$. Therefore, the set of all possible placements (P) can be represented as $P = \{P_1, P_2, P_3, ..., P_m\}$, where m denotes the variations of n elements taken in groups of k without repetitions. The number of all possible placements (m) can be calculated as:

$$m = \frac{n!}{k!(n-k)!} \tag{3.1}$$

"Controller Placement Problem in Software Defined Networking" by Heller [6] was used for reference. In the CPP, one of the most frequently utilized performance metrics is latency (or delay). Generally, latency describes the total time taken by a packet from the source node to the destination node. In this work, we take the latency as the distance between two nodes, and study two types of optimization metrics: average-case latency and worst-case latency [6].

3.4.1 Average-Case Latency

Average latency for the placement of the controller minimizes the average distances between the controller and switches assigned to the controller. This optimization problem is known as the minimum k-means problem [67]. The objective function can be defined as the follows:

$$L_{cs-avg}(P') = \frac{1}{n} \sum_{v \in V} \min_{(c \in P')} d(v,c) \tag{3.2}$$

Where $d(v,c)$ is the distance between switch $v \in V$ associated with controller $c \in C$. The objective is to find the optimal placement of controllers P' from the set of all possible controller placements where $|P'| = k$ and the $L_{cs-avg}(P')$ is the minimum.

3.4.2 Worst-Case Latency

The worst-case latency minimizes the worst (or maximum latency) between a switch $v \in V$ and its controller $c \in C$. This optimization problem is known as minimum k-center problem [99]. The k-center problem is a location analysis problem related to the optimization problem in the area of operation research. The k-center scheme provides an optimal solution to minimizing the longest distance among nodes. The objective function is defined as follows:

$$L_{cs-worst}(P') = \max_{v \in V} \min_{c \in P'} d(v,c) \tag{3.3}$$

The optimization result comes from minimizing the previous optimization function. The goal of the optimization is to find the optimal minimum-latency placements P' from the set of all possible placements P.

3.4.3 Formulation Constraints

The mathematical constraints are computed as follows:

$$\forall i \in V : \sum_{k \in P} x_{i,k} = 1 \tag{3.4}$$

$$\sum_{k \in P} w_k = k \tag{3.5}$$

$$\forall i \in V, \forall k \in P : x_{i,k} \leq w_k \tag{3.6}$$

$$\forall k \in P : w_k \in \{0,1\} \tag{3.7}$$

$$\forall i \in V, \forall k \in P : x_{i,k} \in \{0,1\} \tag{3.8}$$

Equation (3.4) ensures that each switch is associated with exactly one controller. Equation (3.5) restricts the number of controllers deployed exactly to k. Equation (3.6) signifies that switch i is located to controller k if controller k deployed on switch i. Constraints (3.7) and (3.8) specify that the variables $x_{i,k}$ and w_k are binary (equal to 0 or 1). Notations used in the formulation are explained in Table 3.6

Symbol	Description
$G(V,E,L)$	Physical Network
V	Set of switches in the network (nodes)
E	Set of physical links between switches (edges)
L	Set of switch locations
C	Set of Controllers, where ($c \subset V$)
(i,j)	Link between node i and node j
$d(v,c)$	Distance between switch $v \in V$and controller $c \in C$
n	Total number of switches or nodes in the network, where ($n = \|V\|$)
k	Total number of controllers to be installed on the network, where ($k \leq n$)
P	Set of all possible placements for k controllers
w_k	Indicates whether a controller is deployed at location k (= 1) or not (= 0)
$x_{i,k}$	Indicates whether switch i is connected to controller k (= 1) or not (= 0)

Table 3.6 Notations used for controller placement

Chapter 4

Multi-level Hierarchical Controller Placement

4.1 Introduction

The key idea of SDN is separation of the forwarding plane (or data plane) and the control plane to manage network performance. A logically centralized control plane supports SDN feasibility [33], [55]. The focus of this study is to minimize the control plane latency, which comprises transmission delay, propagation delay, switch queuing latency, and controller processing latency [133], [134]. The transmission delay at the switch and controller refers to the ratio of packet size and the data rate of a link. The propagation delay depends on the distance between switches and controllers. The queuing latency is mainly determined by congestion in the path between source and destination. The load of the controller principally influences the controller processing latency. In this research, the metric of the optimization study is based on propagation delay, referred to as latency.

The Controller Placement Problem (CPP) is a non-deterministic polynomial NP-hard problem similar to the facility location problem [135]. Heller et al. [6] proposed the principal objective to solve the CPP by minimizing the propagation delay (latency), in order to compute the optimum location of the controllers deployed and the minimum number of controllers to be placed. The authors conducted experiments on the Internet2 OS3E network topology [136]. K-means is defined to minimize average propagation delay [67], whereas K-center is defined to minimize the largest distance latency between controllers and their associated switches [99]. To find the optimum controller positions of the networks, the following questions must be solved:

- What should be the minimum number of controllers to be placed on a network?

- Where should the controllers be placed?

In this study, we consider the scenario of Uncapacitated Controller Placement Problem (UCPP) [2]. This means that the capacity is not considered as a constraint and the controller is located with a switch [116].

Deploying a single controller may be inefficient for managing a large network and is a single point of failure. Placing multiple controllers working as a single logical controller is still an outstanding challenge. For large sized networks, multiple controllers are more efficient for handling control plane traffic, in order to improve network scalability and latency [56]. Kuang et al. [5] proposed an algorithm based on a hierarchical K-means algorithm to solve a controller placement problem.

The objective of this study is to minimize the maximum latency between the controller and its associated switches and to balance the load of each controller. The main contribution of this work is: (a) the large network is divided into several single controller domains; and (b) in the process of network partitioning, load balancing is taken into account, in addition to latency between switches and controller. The experimental results illustrate the partitioning used the K*-mean algorithm is proven to be more balanced than the optimized K-means algorithm, but the propagation latency is larger than the optimized K-means algorithm by Wang et al. [35].

For very large networks composed of several Autonomous Systems (AS), we explore the feasibility of a Multi-level Hierarchical Control Plane, where controller latencies are considered along with the controller-switch latencies when evaluating the control plane reactivity perceived by switches. The traffic on the controller plane is crucial to achieve a consistent shared view of the network state that is the required condition to run network applications, and the network state is stored in shared data structures. The response time of packets transmitted from the controller to each associated switch, or from the switch to the controller, is usually shorter than in the controller-controller traffic communication.

In general, the controller-controller traffic is very complex, because distributed controllers adopt coordination protocols and algorithms to synchronize their shared data structures to guarantee a consistent global network view and to enable a centralized view of the network state for the applications. The controller-switch traffic in the control plane fundamentally depends on the network application running on the controller. As a result, the optimal controller placement in a given network should consider both kinds of latencies. On the other hand, our work concentrates on the controller placement problem by studying the impact of both latencies.

4.2 Proposed Architecture and Methodology

The proposed Hierarchical Controller Placement Problem (HCPP) methodology adopts a hierarchical architecture, whereby the controller at the higher-level manages the controllers at the lower level. There are at least two levels of controllers and it splits the control plane into multiple levels. The advantage of multi-level architecture is that it improves network scalability and efficiency [35]. Ideally, a hierarchical approach may support any number of levels, but for each scenario, the optimal number should be found. Figure 4.1 illustrates the Multi-level Hierarchical Control Plane Architecture with three controller layers, where the Super Controller (SC) is at the top-level, some Master Controllers (MCs) are at the intermediate level, and the Domain Controllers (DCs) are at the bottom level. There are five different types of communications: Domain Controller and Switch (DC-S), Domain Controller and Domain Controller (DC-DC), Master Controller and Domain Controller (MC-DC), Master Controller and Master Controller (MC-MC), and Super Controller and Master Controller (SC-MC). This architecture requires three controller layers and thus the optimization of three-phases, as described below.

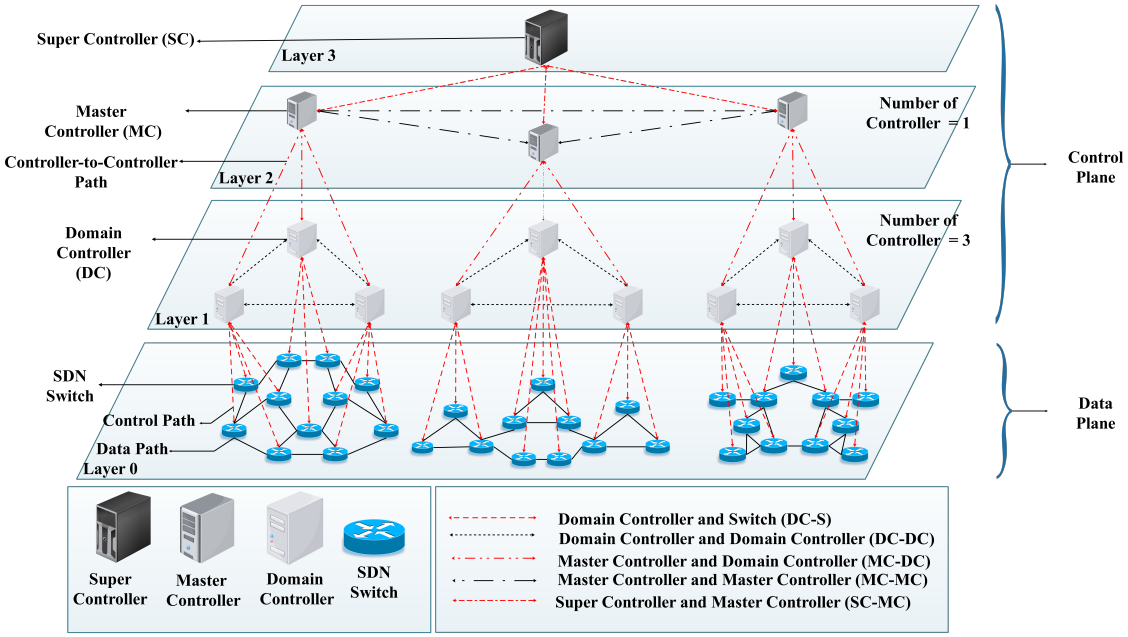

Fig. 4.1 Multi-level Hierarchical Control Plane Architecture

4.2.1 Domain Controllers (Bottom Level)

DC placement optimization minimize latency between switches and DCs. All DCs are at one level, and are responsible for managing switches in the domain.

4.2.2 Master Controllers (Intermediate Level)

MC placement identifies the number and placement of MCs, whereby the DCs act as switches (controlled devices). The topology for connecting DCs is built as a virtual network (VN) based on the physical underlying topology and computing the shortest path between DC pairs. Dijkstra algorithm is implemented on the top physical underlying topology to determine the shortest path for all DC pairs.

4.2.3 Super Controller (Top Level)

In the upper-level, an SC acts as a global controller that is a logically centralized control plane. Again, a virtual topology connecting all MCs is built based on the physical underlying topology. Dijkstra algorithm is used to compute the shortest path between all pairs of MCs to find the placement of the SC.

4.3 Evaluation and Results of HCPP Architecture

The results of the practical implementation of the proposed HCPP on Western European NRENs topology is discussed in this section. This assessment focuses on latency metrics.

4.3.1 Western European NRENs Topology

The proposed method to solve HCPP is applied on a set of networks of Western European NRENs (278 routers), as shown in Figure 4.2, using data from Internet Topology Zoo [137]. We define each country as a cluster, except countries with less than six nodes, which are integrated with the nearest country. This clustering tries to consider the real constraints derived from each network having its own management agency. The data from Internet Topology Zoo presents seven nodes, which are identified only by the name of the city or a country. We completed the database with corresponding coordinates of the city or the capital city of the country. In order to get the coordinates of the city, we checked with Google Maps and obtained its longitude and latitude. Furthermore, we found out that some of the nodes are exactly in the same geographical location. The optimization results may select either of these locations.

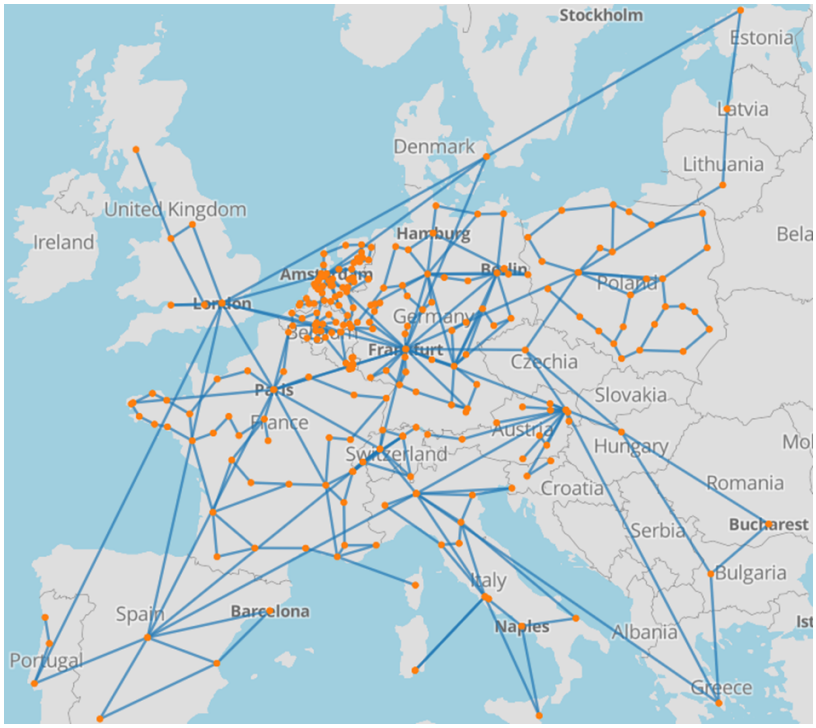

Fig. 4.2 Entire Western European NRENs Network Topology

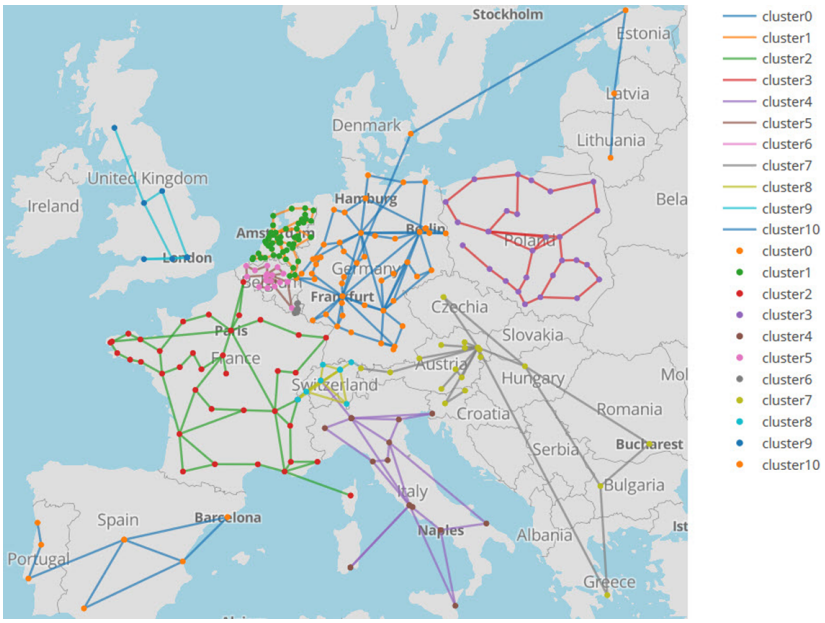

Fig. 4.3 Western European NRENs Network Topology with Clusters

The Western European network is divided into eleven clusters by using a country-based approach (see Figure 4.3). The clusters are listed in Table 4.1. The linearization approach is used to convert the problem into Mixed-Integer Programming (MIP), which is then solved optimally with the well-known optimizer tool IBM ILOG CPLEX Optimization Studio V12.8.0 [112]. CPLEX is used to provide all optimal solutions and the model is programmed in Python. The computing time varies from tenths of seconds to a few tens of seconds. The optimization running time to minimize the worst-case latency takes more time than the average-case latency but this is not relevant as it is an offline optimization for finding the optimal placement of the controllers before their deployment.

Country	Cluster	Nodes	Network
Germany	0	52	DFN
Denmark	0	1	GEANT
Estonia	0	1	GEANT
Latvia	0	1	GEANT
Lithuania	0	1	GEANT
Netherlands	1	51	SURFnet
France	2	39	RENATER
Poland	3	28	PSNC
Italy	4	20	GARR
Belgium	5	19	BELnet
Luxembourg	6	15	RESTENA
Austria	7	13	ACOnet
Slovenia	7	2	ARNES
Czech Republic	7	1	GEANT
Hungary	7	1	GEANT
Bulgaria	7	1	GEANT
Romania	7	1	GEANT
Greece	7	1	GEANT
Switzerland	8	12	SWITCH
United Kingdom	9	9	JANET
Spain	10	6	RedIris
Portugal	10	3	FCCN
Total	11	278	

Table 4.1 Cluster list

4.3.2 Domain Controller (DC) Placement (Clustering-based)

Latency between DC and switches

Figures 4.4 and 4.5 show the Cumulative Distribution Function (CDF) for all possible combination placements of the controllers for Cluster 1 by utilizing a brute-force approach. The best placement corresponds to the lower latency for each value of k (number of controllers). We use Cluster 1 (Netherlands) as an example. The optimum solution when the number of DCs is one ($k = 1$) given an average latency of ($L_{cs-avg} = 0.336$ ms) and the optimal controller location is in Utrecht. For the worst-case latency is ($L_{cs-worst} = 0.689$ ms) and the placement is in Amsterdam. The latency between DCs and their associated switches decreases gradually as more controllers are added to the network. Table 4.2 summarize the optimum latency and the locations of controllers outcomes of the optimization model for each cluster when the number of DCs is varied from one to three. Running the optimization model is explained in Section 3.4.

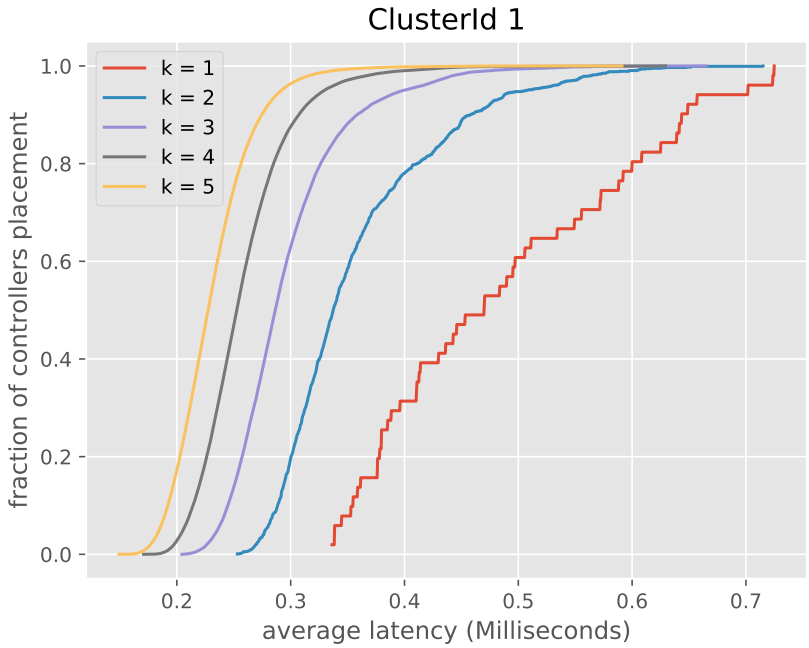

Fig. 4.4 Optimal latency CDFs for all possible controller combination placements for average-case latency

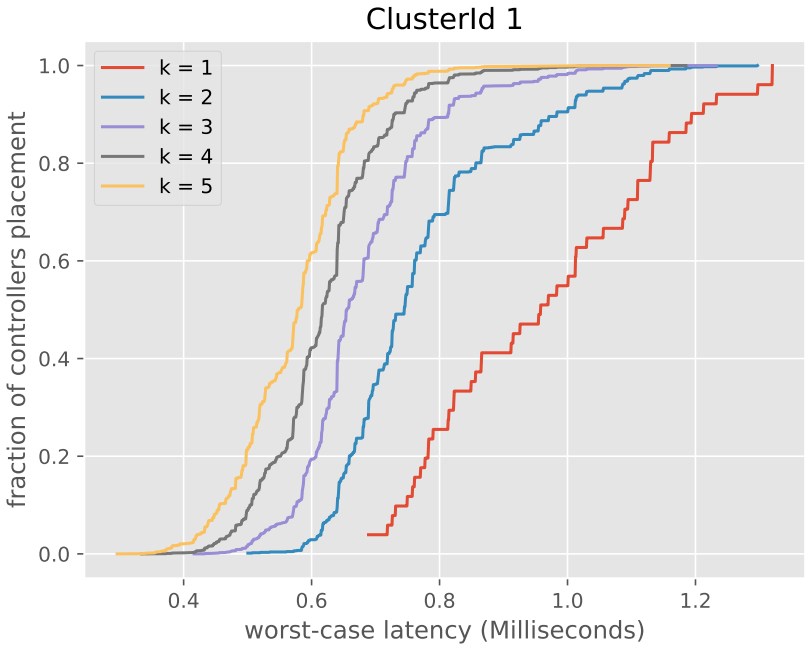

Fig. 4.5 Optimal latency CDFs for all possible controller combination placements for worst-case latency

Benefit to cost ratio

The goal of this metric is to decide if more controllers should be deployed. Adding a new controller reduces latency but adds cost. The Benefit to Cost Ratio is defined as:

$$(lat_1/lat_k)/k \tag{4.1}$$

Where lat_1 represents the latency with a single controller, lat_k represents the latency with k controllers. The reason for dividing by k is to account for the cost of adding controllers [6]. Figures 4.6 and 4.7 show the Benefit to Cost Ratio of deploying DCs when the number of controllers increases from 1 to 12.

In this study, we define a rule to make an easier way to determine how many controllers we need. The minimum number of controllers required corresponds to the latency $(lat1)/2$. Looking for a reduction of the latency when $k = 1$ to a half, we conclude that three to four controllers are suitable to optimize the average latency and four to five controllers for worst-case latency. We found that the suitable number of controllers is similar to the other clusters. Table 4.2 shows the optimal domain controller location for average-case latency and worst-case latency.

Table 4.2 Optimal domain controller location for average-case latency and worst-case latency

Clusters	k = 1	k = 2	k = 3
Average-case latency			
C0	Frankfurt 1.170	Frankfurt, Potsdam 0.883	Frankfurt, Potsdam, Riga 0.676
C1	Utrecht 0.336	Hoogeveen, Utrecht 0.253	Hoogeveen, Utrecht, Breda 0.204
C2	Paris 1.494	Nantes, Lyon 1.080	Paris, Nantes, Lyon 0.783
C3	Lodz 1.047	Gdansk, Krakow 0.780	Gdansk, Radom, Gliwice 0.599
C4	Bologna 1.138	Cagliari, Milan 0.637	Rome, Milan, Cagliari 0.520
C5	Evere 0.257	Evere, Louvain-la-Neuve 0.201	Evere, Louvain-la-Neuve, Liege 0.168
C6	Kirchberg 0.027	Kirchberg, Ettelbruck 0.017	Kirchberg, Esch-sur-Alzette, Diekirch 0.012
C7	Vienna 1.081	Vienna, Sofia 0.740	Vienna, Sofia, Dornbirn 0.631
C8	Lausanne 0.422	Lausanne , Zurich 0.301	Lausanne, Zurich, Kreuzlingen 0.194
C9	London 0.583	London, Warrington 0.289	London, Warrington, Glasgow 0.173
C10	Madrid 1.320	Madrid, Coimbra 0.706	Madrid, Coimbra, Barcelona 0.543
Worst-case latency			
C0	Copenhagen 4.492	Frankfurt, Riga 2.238	Copenhagen, Hannover, Riga 1.762
C1	Amsterdam 0.689	Zwolle, Eindhoven 0.500	Nijmegen, Leeuwarden, Tilburg 0.416
C2	Limoges 3.027	Rouen, Marseille 2.038	Paris, Nantes, Cadarache 1.696
C3	Lodz 1.899	Gdansk, Gliwice 1.369	Gdansk, Lodz, Bielsko-Biala 1.153
C4	Rome 2.300	Rome, Milan 1.788	Cagliari, Rome, Milan 1.256
C5	Vilvoorde 0.655	Antwerpen, Namur 0.406	Antwerpen, Namur, Evere 0.375
C6	Kirchberg 0.090	Hollerich, Diekirch 0.053	Diekirch, Esch-sur-Alzette, Limpertsberg 0.031
C7	Budapest 3.853	Vienna, Sofia 1.750	Vienna, București, Athina 1.715
C8	Lausanne 0.770	Bern, Kreuzlingen 0.506	Bern, Kreuzlingen, Manno 0.484
C9	Warrington 1.179	Warrington, Telehouse 0.986	Glasgow, Warrington, Reading 0.374
C10	Madrid 2.620	Madrid, Lisboa 1.685	Lisboa, Sevilla , Valencia 1.010

Fig. 4.6 Benefit to cost ratio for average-case latency (milliseconds)

Fig. 4.7 Benefit to cost ratio for worst-case latency (milliseconds)

4.3.3 Master Controller (MC) Placement

In the MC placement optimization, we generate the virtual network (VN) topology using the physical underlying real topology interconnecting the DC. The Dijkstra algorithm is applied to compute distances and shortest paths between all pairs of DCs. Figure 4.8 presents MC-DC latency. It can be seen that the optimal average latency gradually decreases with increasing number of DCs. On the other hand, results in the worst-case scenario indicate the instability in some optimal solutions due to the limitation of the data links between two points in the real world topology, despite the fact that others are relatively similar to the average scenario (Figure 4.9).

The results for MC placement optimization show that a single MC provides the best performance on the Western European NRENs topology. The solution of the MC placement gives us a single master controller and it is the optimal solution. This means that we do not need to apply SC placement optimization for this particular topology. The MC is the main controller that manages DCs in different countries, while the DCs manage switches in cities in the same country or neighboring countries.

Figure 4.10 and 4.11 show the particular case where four DCs per cluster are deployed. This means that the total number of DCs is 44 for the MC placements. Then, we build a virtual topology for the 44 DCs. The Dijkstra algorithm is implemented on the top of the physical underlying topology to determine the shortest path between each DC pair. Then, we run the optimization to find the best MC placement.

For the average-case latency, Figure 4.10 shows the MC is located in Frankfurt (Germany). In addition, another significant factor is the massive connectivity between Frankfurt city and other cities in the network. Figure 4.11 illustrates the results of the MC placement for the worst-case scenario. The MC placement is CERN (Geneva, Switzerland).

4.3.4 Applicability Case Study

In this section, we compare the results obtained when applying the hierarchical controller placement against the case of a single level controller placement. The main point is to demonstrate the feasibility of deploying multiple levels of controllers in very large networks composed of autonomous networks, as in the case of the European NRENs. Each autonomous network has its own management policy and coordinates with its neighbors. Considering the case of inter-domain routing policies or end-to-end traffic engineering policies, each network manager wants to have close control over how the policies are applied within its network and, at the same time, wishes to disclose a limited amount of information and details to its

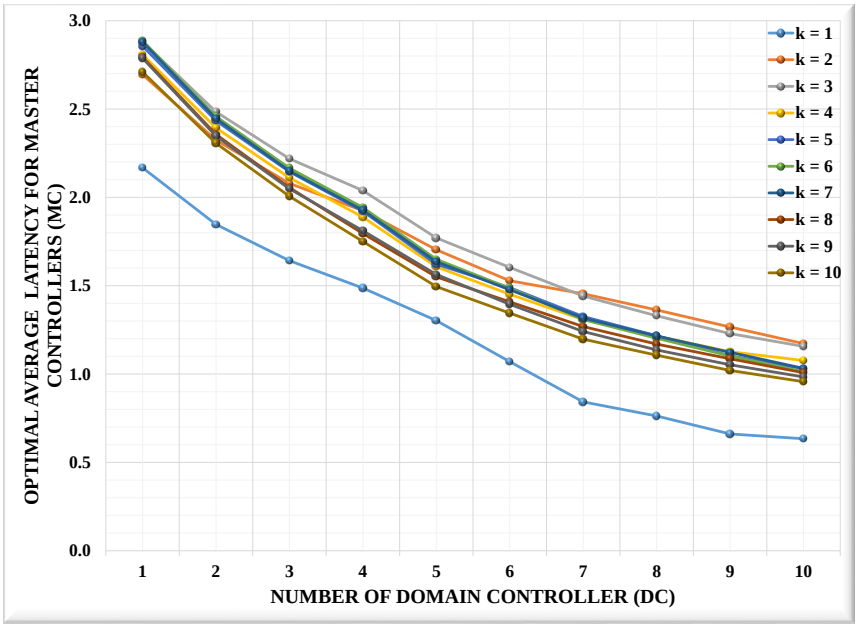

Fig. 4.8 Optimal solution for MC placement for worst-case latency (ms)

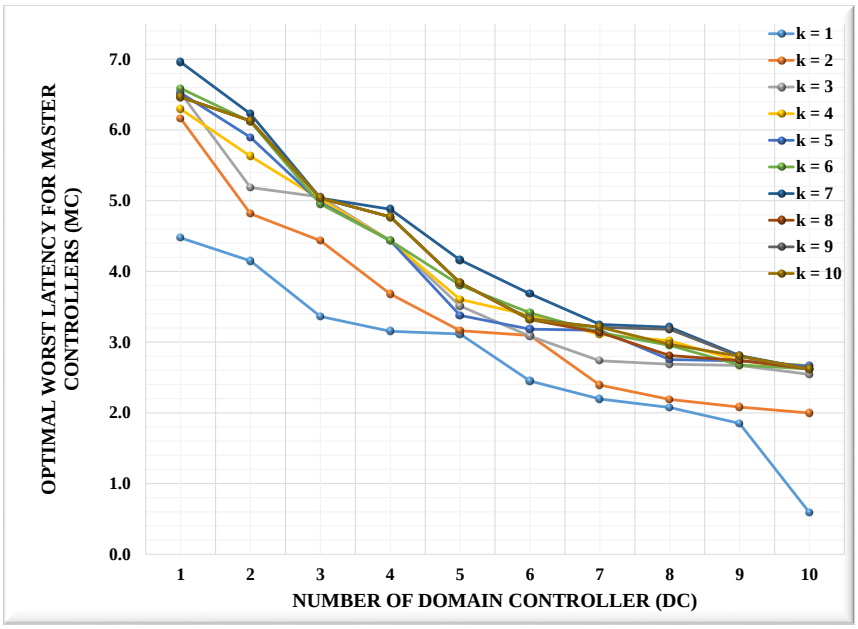

Fig. 4.9 Optimal solution for MC placement for worst-case latency (ms)

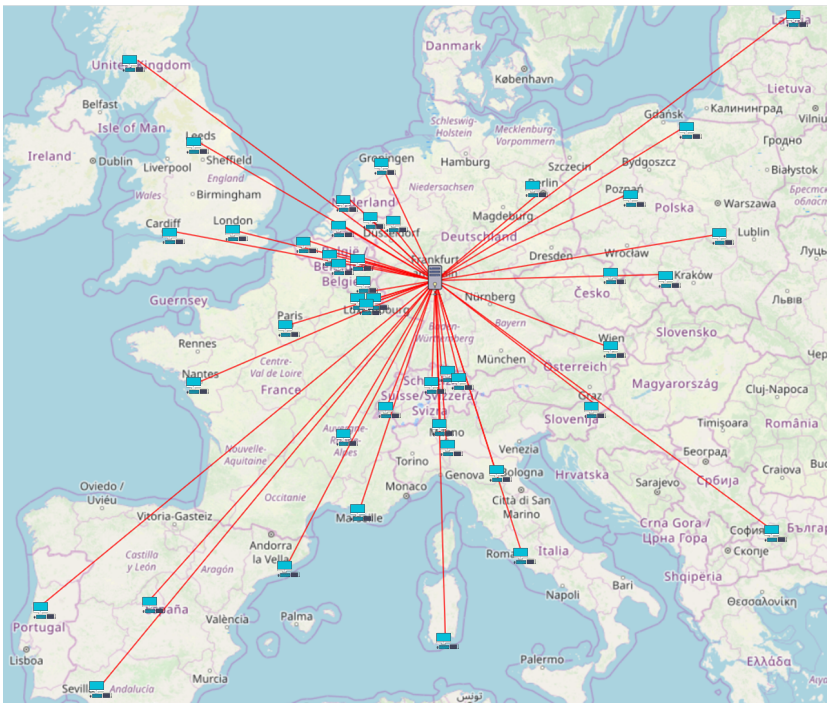

Fig. 4.10 Particular case for MC location for average-case latency

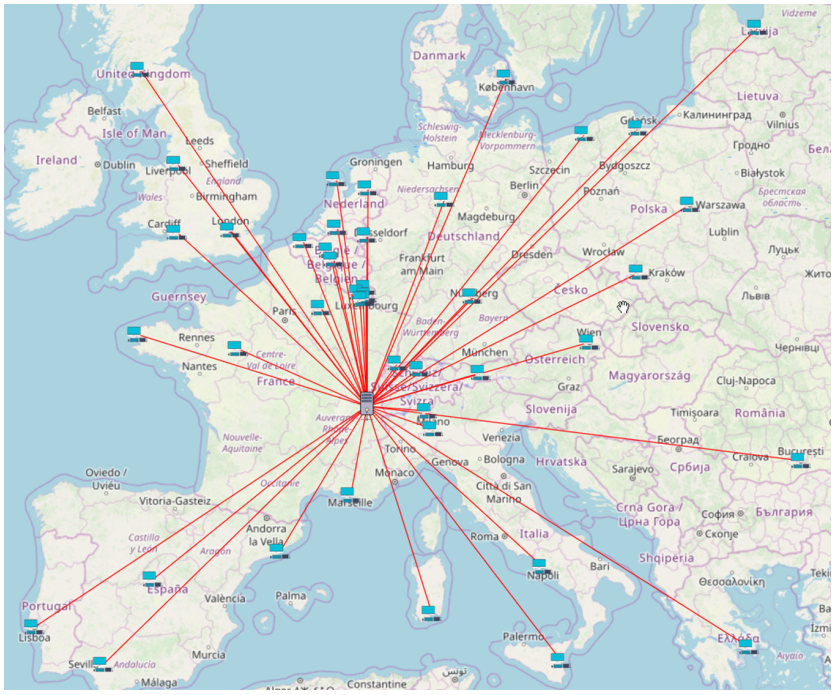

Fig. 4.11 Particular case for MC location for worst-case latency

neighbors. DCs are the appropriate approach for the application of intra-domain rules, while MCs may deal with the inter-domain rules.

For this comparison, we consider the average latency case. The whole set of NREN switches is considered as a unique network and a single optimization step results in the optimal location of the controllers. Following the same optimization function and approach presented in Section 3.4, the average latency of one single controller is 2.27 ms (lat1); and using the same rule to determine the best number of controllers needed (looking for lat1/2) results in six controllers (with an average latency of 1.13 ms). Figure 4.12 shows the related results for optimal average latency of the Whole Western European NRENs without applying the Multi-level Hierarchical Control Plane methodology, when the number of controllers is varied from 1 to 12. Figure 4.13 presents the map with the location of the six controllers [(Vienna, Austria), (Amsterdam, Netherlands), (Frankfurt, Germany), (Milan, Italy), (Paris, France) and (Lodz, Poland)]. This result is referred to as flat control plane.

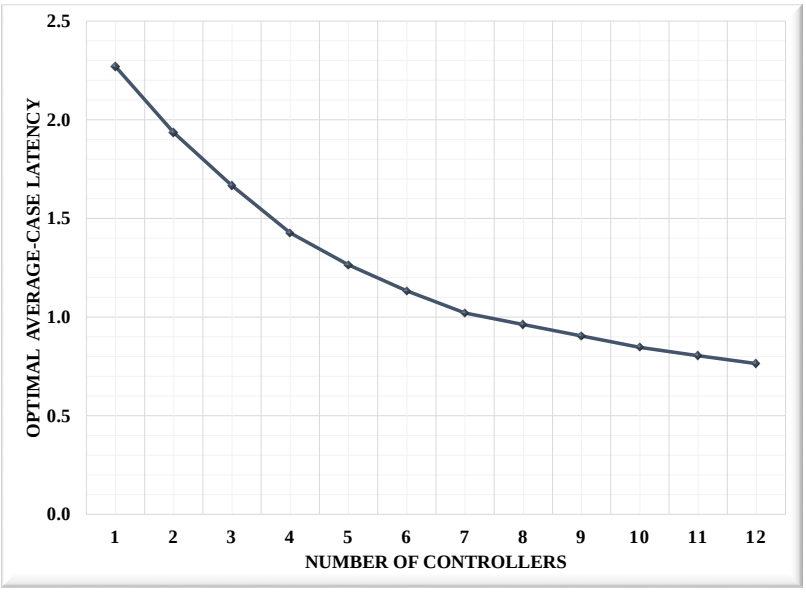

Fig. 4.12 Optimal average latency (ms) of the whole Western European NRENs for flat control case

The results obtained applying the hierarchical method indicate that three DCs per cluster is the best option using the decision rule presented, with MC placement results suggesting a single MC in Frankfurt for the average latency case; this case is referred to as the Hierarchical Control Plane. Comparing the results, for the flat control, the average latency is 1.13 ms. For the hierarchical plane, the average latency between the DC and Switch is about 0.40 ms, and

Fig. 4.13 Whole Western European NRENs with optimal average location when the number of controllers is six (k=6) for flat control case

the average latency between the MC and DC is 1.08 ms. As most decisions are made by DCs, the average latency is better with the hierarchical control, but for some higher-level decisions, the average latency will be about the same than in the flat control approach. This is true only if all information from all the autonomous networks is shared and sent to the controllers in the flat control case.

From a practical point of view and real feasibility, we claim that the hierarchical control case allows detailed control within each domain, and that the extra latency that may occur for some decisions made at the MC will be compensated by the gain in management flexibility and autonomy.

4.4 Chapter Summary

Placement of controllers has received significant attention in recent years in large scale networks. In this study, we minimized the latency between controllers and their associated

switches, presenting a Multilevel Hierarchical Control Plane Architecture. The applicability of this technique is evaluated using real-world Western European NRENs. The topologies are taken from the Internet Topology Zoo collection. Up to our knowledge, this is the first kind of controller placement study applied to the NREN in Europe.

The design methodology was performed from the bottom to the top because we consider the administrative boundaries; each NREN has its own management policy, which is a fundamental constraint. For this reason, the real network topology (Western European NRENs) was divided into clusters by using a country-based approach. The main contributions are the following. As a result of the DC placement optimization, we provide a rule to make an easier way to determine how many controllers are needed. As a result, we found out that the number of DCs is similar for each cluster based on the K-means and K-center metrics. Generally, three to four DCs are required in case of average latency; or four to five DCs in the worst-case latency scenario.

Second, the MC placement proposed during the MC placement optimization showed that a single MC is sufficient to manage the entire network to achieve the best performance in respect of this particular case study.

The overall contribution is the methodology presented to define a multi-level control plane as an iterative optimization problem. For this particular topology, the results show that a third level is not needed. Other large topologies with different constraints may need a third level in the control plane.

All datasets and results will be made available for research purposes. Further research is in progress to consider the reliability and capacity constraints of the Hierarchical Controller Placement Problem (HCPP).

Chapter 5

Joint Latency Controller Placement of Control Plane

5.1 Introduction

SDN is a programmable network architecture that decouples the control plane (controller logic) from the data plane (forwarding plane) to provide flexible network management. A logically centralized control plane supports SDN feasibility. A single physical controller may not be capable of controlling the whole network. To tackle this issue, deploying multiple controllers is a necessity to handle a large network. CPP is one of the most important issues in SDN to improve scalability, and it influences the performance of the whole network. Most previous methods only focused on latency between controllers and their associated switches, ignoring the latency between controllers and the accumulated latency of the control plane. In this study, we define an accumulated latency to solve controller placement, which takes into consideration both the latency between the controller and their associated switches and the inter-controller latency. We formulate an optimization problem as a MILP under the constraint of latency, with the objective of minimizing the accumulated latency and minimize the number of network controllers while optimizing their placement, to achieve an optimal balance at the same time. The performance of our method is evaluated on the Internet2 OS3E real network topology. Results demonstrate that the proposed method is promising.

In distributed SDN controllers, two control planes are identified (as shown in Figure 5.1). First, we consider the control traffic exchanged between controllers and their associated switches (CS plane). Second, we focus on the control traffic exchanged among controllers (or inter-controller) (CC plane), which supports the interaction between controllers (i.e., the control traffic to keep the shared data structures synchronized). We focus our research on the

latency tradeoff achievable in the CS and in the CC control planes. The interaction between CS plane corresponds to the southbound interface, i.e., OpenFlow, and the interaction between CC plane and the data stores corresponds to the east-west interface.

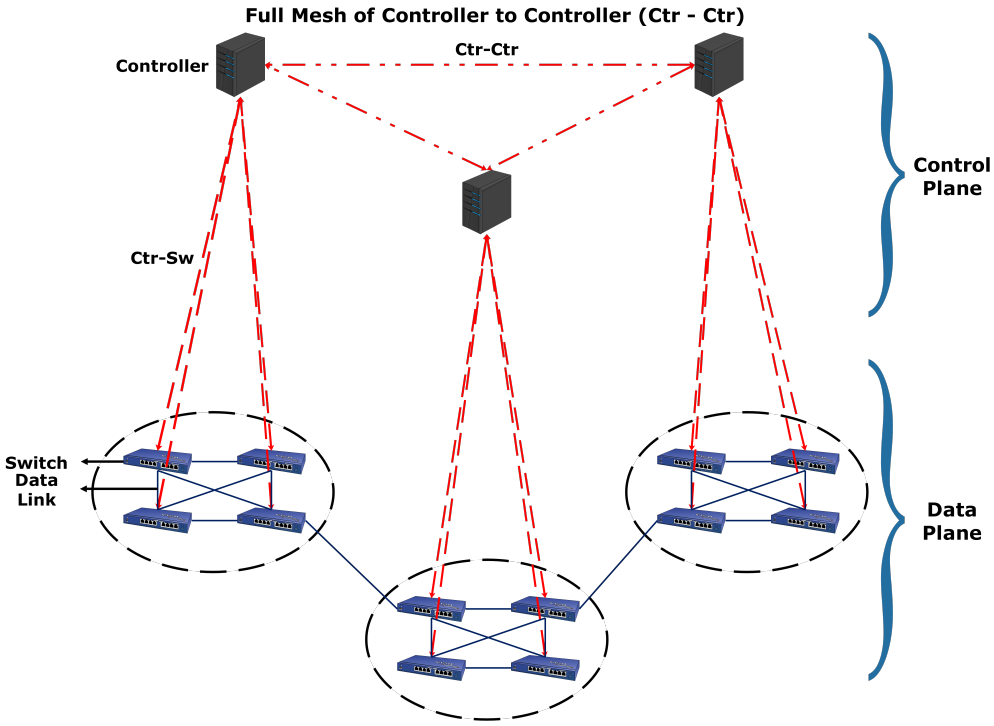

Fig. 5.1 The Architecture of SDN Control Plane

For communication within the control plane, the inter-controller traffic plane is important to achieve a consistent shared view of the network state, which is the essential requirement to run network applications correctly and maintain the network state that is stored in shared data structures. Reducing inter-controller latencies is required to reduce switch and controller latencies, but (for topological reasons) minimizing one increases the other, and vice versa. The placement of the controller influences the control traffic latency. Deploying multiple controllers increases CC traffic, while a few concentrated controllers increase the CS traffic latency.

The CS traffic refers to the packets transferred between the controller and associated switches, or from switches to the controller. Moreover, the reaction time of sending the messages from the controller to each switch traffic is usually shorter than the CC traffic communication. In general, CC traffic is very complex, because controllers may need to synchronize through a consensus algorithm shared data structures to guarantee a consistent global network view of the underlying network and to enable a centralized view of the

network state for the applications. The southbound interface (CS traffic) communication is faster than the east-west interface (CC traffic) communication. The optimal placement of the controllers must not only consider the latencies of CS traffic, but also the latencies between controllers. Depending on the application of the control plane, it may be more important to minimize the CC latency than CS traffic communication. For each application running on the controller plane, we can assign the weight between CS and CC latencies to determine the optimal placement of the controllers.

This work considers the joint optimization of latency and controller placement of both the CS (southbound) interface and CC (east-west) interface. The main objective of this study is to minimize the control plane latency, which takes into consideration both latency from the controller to switch and controller to the controller simultaneously. The remainder of this chapter is structured as follows. The related work in the CPP is presented in Section 5.2. The mathematical model and formulation is described in Section 5.3. The evaluation results are discussed in Section 5.4. Finally, Section 5.5 summarizes the main conclusions from this chapter.

5.2 Motivation and Related Work

In this section, we briefly discuss the most related works that addressed the controller placement problem in SDN. The CPP is a key design choice of the SDN control plane to increase performance. The problem is similar to the facility location problem, which can be denoted as a non-deterministic polynomial NP-hard problem [135]. Heller et al. [6] first improved the scalability of multi-controller for solving CPP, to determine the optimum location of the controllers deployed and the minimal number of controllers to be placed. Internet2 OS3E topology has been used to evaluate performance [136]. The placement of the controller directly influences the latency between switches and associated controllers, thus affecting the performance of the whole network. To solve the CPP, the authors proposed an optimized algorithm based on the K-means and K-center algorithms. The K-means algorithm is designed to minimize the average propagation delay between the controllers and their assigned switches [67], while K-center is designed to minimize the maximum latency between controllers and switches [99].

Earlier solutions of CPP only focused on propagation latency between controllers and their associated switches and inter-controller latency metrics were ignored. Hock et al. [116] proposed a framework for resilient Pareto-based Optimal Controller Placement (POCO) that provides the operator of a network with all Pareto-optimal placements. In their paper, the load on the controllers and inter-controller latency are considered. The POCO provides better

load balancing between switches and its controllers and balances the placement of controllers. If a large number of controllers are required for managing the network, synchronization is necessary to maintain a consistent global state. Depending on the frequency of the inter-controller synchronization, the latency among controllers plays an important role and thus should be considered during the controller placement.

Gao et al. [138] introduced a framework for the Global Latency Control Placement Problem with Capacitated Controllers (CGLCPP). Global latency is the combination of the switch to controller latency and inter-controller latency. The optimization objective is to minimize global latency on random network topologies, which takes into consideration the latency between controllers and switches, the latency between controller to controller and the capacities of controllers. In their theoretical research paper, they proposed a PSO algorithm to solve the controller placement in SDN, but they did not consider the Reliable Controller Placement Problem (RCPP). The PSO-based solution provides better performance in terms of latency and computational time as compared to Integer Linear Programming (ILP) and greedy algorithms.

Wang et al. [139] implement Network Clustering-based Particle Swarm Optimization Algorithm (NCPSO) to optimize the controller load under propagation delay and load-balancing constraints. The authors extended the previous work of Gao et al. [138]. They also partitioned the network into k small domains, each with its own controller for control and management, and they considered the load of controllers, switch to controller latency, inter-controller latency, and load balancing. Their simulation results demonstrated the effectiveness of the proposed algorithm, which can significantly reduce the number of required controllers, load of the maximum-load controller, and load balance. They randomly generated some different sizes of topology. The results showed better performance compared to K-center and capacitated K-center strategy [6], [88].

RCPP was defined with the implementation of PSO (Particle Swarm Optimization) algorithm to solve CPP and TLBO (Teaching-Learning Based Optimization) to solve the RCPP in the network, to minimize the total average latency of a reliable network [140]. Simulations were tested for the two most popular topologies (Internet2 with 34 nodes, and Savvis with 19 nodes), revealing that the solution of TLBO gives better performance as compared to the PSO based solution for CPP.

In our study, we defined a new metric for the joint optimization of latency taking into account the CS latency and CC latency simultaneously. The objective of this study is to find the optimal placement of the controllers that minimizes the accumulated latency in the control plane. As well in this work, we focus exclusively on minimizing the number of

controllers required while optimizing their placement between CS latency and CC at the same time.

5.3 Formulation of Joint Latency Controller Placement (LCP) Optimization

The SDN network topology can be represented by a graph $G = (V, E)$, where V represents the set of all switches (or nodes) and E represents the set of all physical edges (or links) between the switches, where the link weight refers to physical distance. We denote the set of switches as $V = \{v_1, v_2, ..., v_n\}$, where n represents the total amount of nodes ($n = |V|$). The set of controllers to be installed is represented as $C = \{c_1, c_2, c_3, ..., c_k\}$, where k represents the number of controllers to be deployed throughout the network ($k \leq n$). Let M denote the distance matrix between all edges and $dist_{(i,j)}$ denote the distance from the node i to node j. A matrix consists of the shortest path latency. For each index (i, j), it corresponds to the shortest path latency from node i to node j. A node may be a simple switch or a controller collocated with the switch. The details of the notation are summarized in Table 5.1.

Notations	Definition
$G(V,E)$	Graph G, where V is a set of switches and E is a set of edges between switches
V	Set of switches or nodes in the network
E	Set of physical links or edges between switches
C	Set of controllers to be installed, where ($C \subset V$)
$dist_{(i,j)}$	Distance from node i to node j
(i, j)	Link from node i to node j
n	Total number of nodes in the network, where ($n = \|V\|$)
k	Total number of controllers to be installed on the network, where ($k \leq n$)
$x_{i,j}$	Indicate whether switch i is mapped to controller j (= 1) or not (= 0)
$y_{i,j}$	Denotes whether controller k is placed on switch i (= 1) or not (= 0)
$n_{(CC)}$	Number of possible links between controllers is $k(k-1)/2$
CS	Controller-switch latency
CC	Controller-controller (inter-controller) latency

Table 5.1 Notations used for Joint Latency Controller Placement (LCP) Optimization of the control plane

The Joint Latency Controller Placement (LCP) Optimization model is based on the original CPP model [6], with some additional constraints and terms for the objective function. More particularly, we model the Joint Latency Controller Placement (LCP) to find an optimal trade-off between switch-controller latency and inter-controller latency. The goal is to minimize the accumulated latency of the network, mathematically formulated as:

$$\min(\lambda * CS + (1-\lambda) * CC) \tag{5.1}$$

The first part of the objective function considers the latency between switches and controller (CS weighted by λ) and the second part of the function considers the inter-controller- latency (CC weighted by 1-λ). In other words, the accumulated latency is the weighted combination of switch to controller latency and inter-controller latency. The objective is to determine the optimal number and location of controllers to achieve a given balance between controllers and switch latency, and controller to controller latency in the control plane.

In this work, we consider the latency as the distance by using Dijkstra's algorithm for computing the shortest path distance between pairs of nodes. Furthermore, we consider two types of optimization metrics: average-case latency and worst-case latency. The controller is collocated with a switch and each switch is managed by a single controller.

Latency between controllers and switches (CS)

- Average-case latency scenario. The average latency between controllers and their assigned switches is computed as:

$$CS = \frac{\sum_{i \in V, j \in C} dist_{(i,j)} x_{i,j}}{(n-k)} \tag{5.2}$$

  Where $dist_{(i,j)}$ is the shortest distance path from switch i to controller j and n represents the total amount of nodes in the network and k represents the number of the controllers. This optimization problem is known as the minimum k-means problem.

- Worst-case latency scenario. The objective is to minimize the maximum latency between the controller and their associated switches.

$$CS = \max_{j \in C} \left(dist_{(i,j)} x_{i,j} \right), \forall i \in V, j \in C \tag{5.3}$$

  This optimization problem is known as the minimum k-center problem.

Inter-controller communication latency (*CC*)

- Average-case latency scenario. The average inter-controller latency is minimized. The number of possible paths among controllers is highly dependent on the number of controllers k is calculated as $n_{(CC)} = k(k-1)/2$. In order to compute the average latency, the following equation is used:

$$CC = \frac{\sum_{i \in V, j \in C} dist_{(i,j)} x_{i,i} y_{j,j}}{n_{(CC)}} \tag{5.4}$$

- Worst-case latency scenario. To solve the worst-case inter-controller latency, the longest distance between controllers is computed. The optimization problem is to minimize the maximum latency among controllers. The considered metric is formulated as:

$$CC = \max_{j \in C} \left(dist_{(i,j)} x_{i,i} y_{j,j} \right), \forall i \in V, j \in C \tag{5.5}$$

The constraints of this model are formulated as:

$$\sum_{j \in C} x_{i,j} = 1, \forall i \in V \tag{5.6}$$

$$\sum_{i \in V} y_{i,j} = 1, \forall j \in C \tag{5.7}$$

$$y_{i,j} \leq x_{i,j}, \forall i \in V, \forall j \in C \tag{5.8}$$

$$x_{i,j}, y_{i,j} \in \{0,1\} \tag{5.9}$$

Constraint 5.6 guarantees that every switch is assigned to exactly one controller. Constraint 5.7 ensures that each controller is located on exactly one switch. Constraint 5.8 ensures that switch i is mapped to the controller j if controller j is placed on switch i. Constraint 5.9 guarantees that the decision variables are binary (equal to 0 or 1).

5.4 Evaluation Results

The goal of the model is to minimize the accumulated latency of the control plane, which takes into consideration the latency between controllers-switches and inter-controller simultaneously. Therefore, the placement of controllers may vary according to the parameter λ. The real network topology used is the Internet2 OS3E network, which has 34 nodes and 42 links. The mathematical models are implemented in Python. We run the optimization as a MILP

model. Then, we solve the problem by using Gurobi Optimizer [111], one of the fastest and most powerful solvers.

5.4.1 Sensitivity analysis of lambda

When the weight of λ increases, the objective function is more focused on the latency between controllers and their associated switches and when the λ value decreases, the objective function is more focused on the inter-controller communication latency.

Figures 5.2, 5.3 and 5.4 show how different values of λ influence the optimal placement of the controllers for 5 controllers. The placement of the controllers in the network is of paramount importance, and implies different trade-offs between the switch to controller latencies and controller to controller latencies.

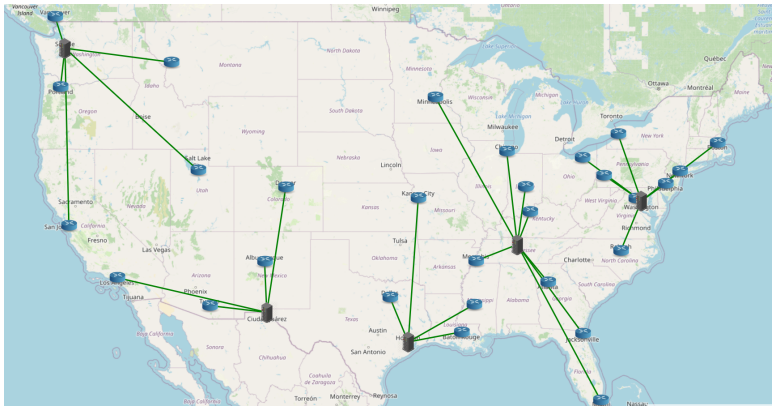

Fig. 5.2 Optimal controller location ($\lambda = 1$ and $k = 5$)

Fig. 5.3 Optimal controller location ($\lambda = 0.9$ and $k = 5$)

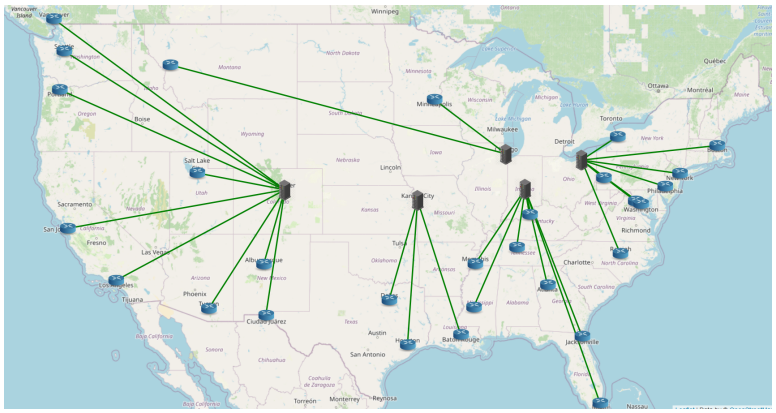

Fig. 5.4 Optimal controller location ($\lambda = 0.8$ and $k = 5$)

Figure 5.5 depicts the results of the accumulated latency of average switch-controller latency and average inter-controller latency when the number of controllers varies from 1 to 12 and the values of λ range from 0 to 1.

When the weight of λ is one, we optimize the results for the accumulated latency between switches and their controllers only. This case corresponds with the basic CPP optimization [5]. Each switch selects the nearest controller in the network. For 5 controllers, the optimal average latency is 1.683 ms and the optimal placement of the controllers is Seattle, El Paso (Texas), Nashville, Houston and Ashburn (Virginia). Results show that the average latency between controllers and switches can be reduced by increasing the number of controllers.

When the value of λ is zero, the optimization considers the inter-controller latency only, and all the controllers are at the closest distance. In this case, because the model does not consider the switches and controllers' latency, some switches might be far from their assigned controller. Increasing the number of controllers, the accumulated latency sharply increases. For 5 controllers, the optimum accumulated latency is 2.022 ms, and the controller location is New York, Philadelphia, Washington DC, Pittsburgh and Ashburn (Virginia).

When the value of λ is 0.5, it means that a balance between controller-switch latency and inter-controller latency is sought. For 5 controllers, the optimal placement of controllers is Chicago, Indianapolis, Louisville, Nashville and Memphis and the average accumulated latency is 3.758 ms.

For 5 controllers when $\lambda = 0.8$, the minimum accumulated latency is 3.690 ms, and the optimal controller location is Denver, Chicago, Kansas City (Missouri), Indianapolis and Cleveland (see Figure 5.4).

It is worth noting that when the number of controllers deployed reaches a certain value in the network, the accumulated latency begins to increase as the number of controllers

increases. This is because when the number of controllers increases, the inter-controller latency increases rapidly.

This method can be applied under different scenarios, including: average CS latency and worst-case CC latency (Figure 5.6), worst-case CS latency and average CC latency (Figure 5.7), worst-case CS latency and worst-case CC latency (Figure 5.8).

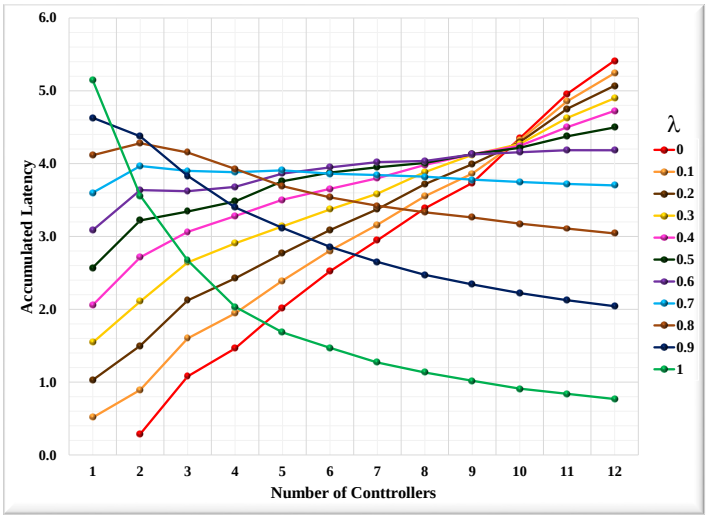

Fig. 5.5 Accumulated latency of average switch to controller latency and average inter-controller latency

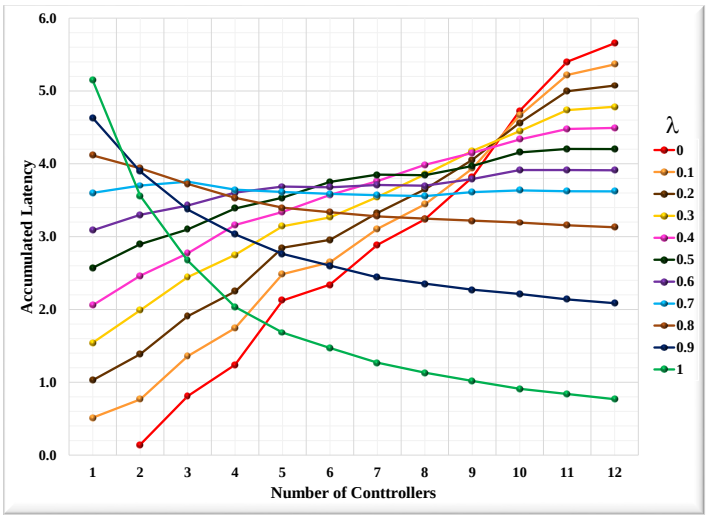

Fig. 5.6 Accumulated latency of average switch to controller latency and worst-case inter-controller latency

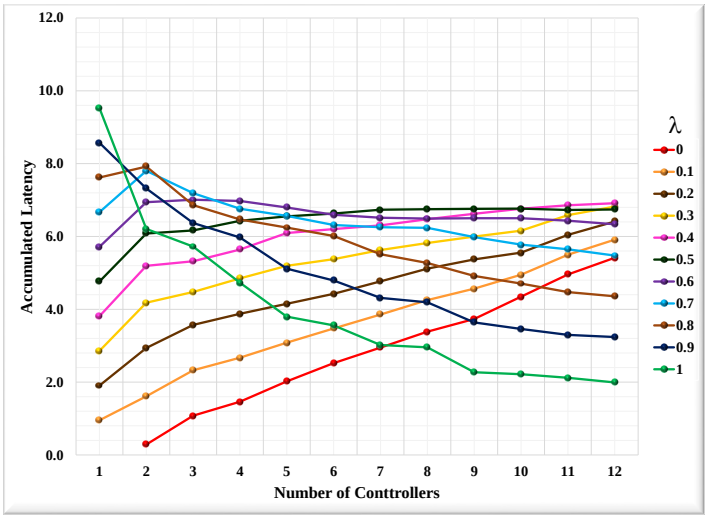

Fig. 5.7 Accumulated latency of worst-case switch to controller latency and average inter-controller latency

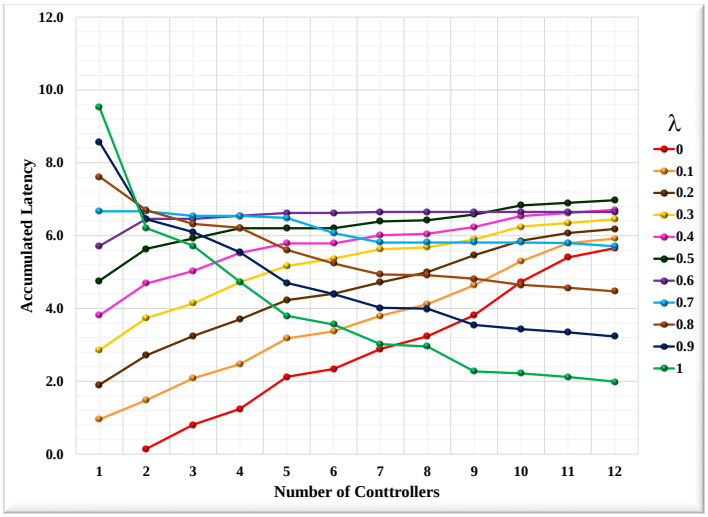

Fig. 5.8 Accumulated latency of worst-case switch to controller latency and worst-case inter-controller latency

5.4.2 Balancing switch-controller and inter-controller latency

In the first analysis, we define the weight for controllers to switches and inter-controllers by setting the value of λ. For each control plane application, a value of λ is set and the placement of the minimum number of controllers can be found. In this subsection, we assume

that a balance between CS and CC latency is also a goal to achieve the required performance of the control plane.

In the next step in our work, we define a decision rule that provides a balance between controllers and their associated switches and inter-controllers simultaneously. The difference between CS and CC latency is seen in Table 5.2 as follows:

$$(abs(CS - CC)) \tag{5.10}$$

k / λ	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
k =2	6.07	6.07	6.07	6.07	4.16	4.16	3.13	3.13	0.07
k = 3	5.22	5.22	5.22	2.79	2.79	2.79	2.79	0.02	3.46
k = 4	4.81	4.81	4.81	2.01	2.01	2.01	2.01	1.04	7.69
k = 5	3.72	3.72	3.72	3.12	1.05	1.05	1.01	4.25	10.73
k = 6	2.83	2.83	2.83	2.83	0.92	0.47	1.35	4.07	10.78
k = 7	2.13	2.13	2.13	2.13	0.73	0.37	4.22	4.22	13.00
k = 8	1.66	1.66	1.66	0.27	0.27	0.27	4.51	5.08	11.88
k = 9	1.28	1.28	1.28	0.02	0.02	0.81	5.18	5.30	11.64
k = 10	0.26	0.26	0.26	0.26	0.26	1.98	5.22	7.25	12.03
k = 11	1.01	1.26	1.26	1.26	1.26	2.76	5.50	7.23	11.40
k = 12	1.72	1.72	1.72	1.72	2.89	4.59	5.73	9.20	11.13

Table 5.2 Difference between average switch-controller latency and average inter-controller-latency (milliseconds)

Figures 5.9, 5.10, 5.11 and 5.12 depict comparative results of overall control plane latency with the minimum number of controllers to provide a tradeoff between CS latency and CC latency, to achieve a balance when the number of controllers varies from 2 to 12 and λ = 0.5. We consider the impact of the number of controllers on the latency performance. The latency between CS decreases with more controllers in the network, while the latency among controllers increases with the increase in the number of controllers.

Figure 5.9 depicts the average CS latency and average CC latency with a different number of controllers. In this case, the balance is achieved with 9 controllers. In the case of the average CS latency and worst-case CC latency, 6 controllers are sufficient to manage the control plane scalability (see Figure 5.10). Figure 5.11 shows the impact of the number of controllers on the worst-case CS latency and average CC latency. Results show the latency

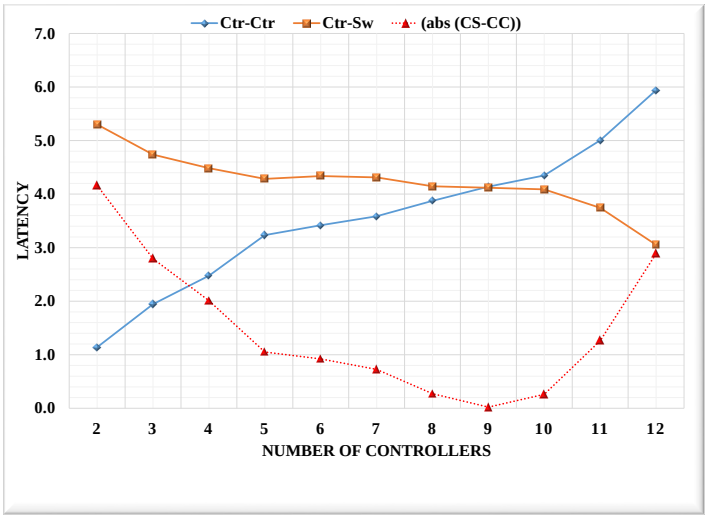

Fig. 5.9 Average switch to controller latency and average inter-controller latency (λ= 0.5)

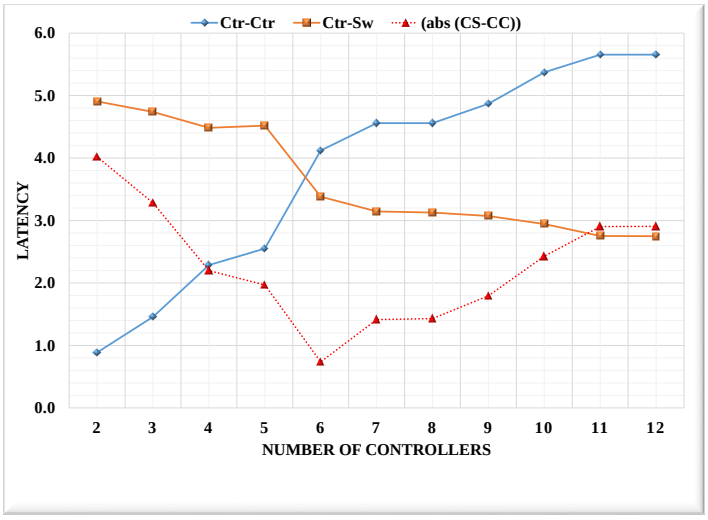

Fig. 5.10 Average switch to controller latency and worst-case inter-controller latency (λ= 0.5)

tradeoff achievable for 7 controllers to perform better performance. Figure 5.12 shows the result of worst-case CS latency and worst-case CC latency. In this case, the optimum number of controllers deployed in the best locations is 12 controllers. Thus, the proposed method is applicable to all other cases with different values of λ and average and worst-case latency.

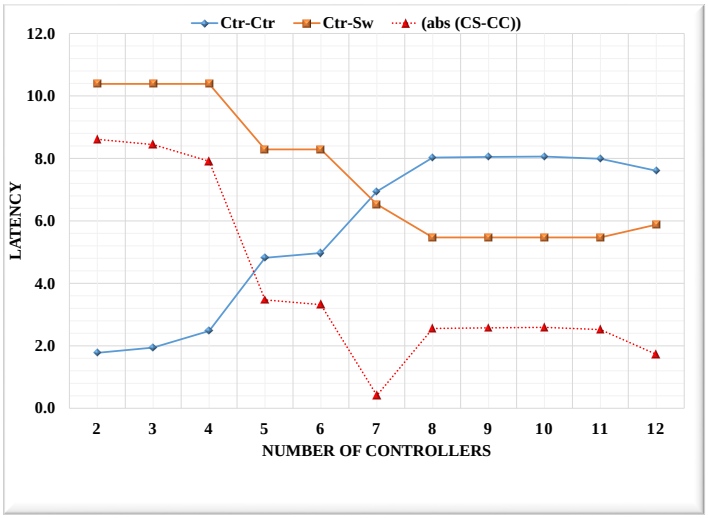

Fig. 5.11 Worst-case switch to controller latency and average inter-controller latency (λ= 0.5)

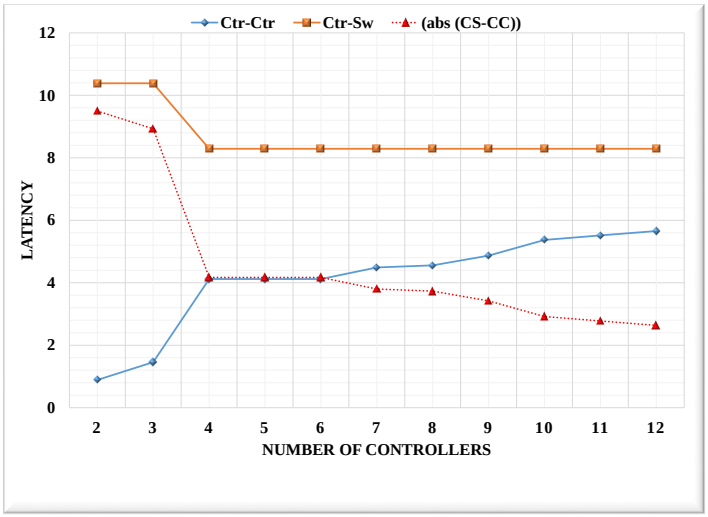

Fig. 5.12 Worst-case switch to controller latency and worst-case inter-controller latency (λ= 0.5)

5.4.3 Control Plane Latency

In this subsection, we define the control plane latency as a sum of CS and CC latency. As illustrated in Figure 5.13, the average-case latency solution gives better results than the worst-case latency solution for the placement of controllers in the control plane. It is interesting to

note that the average CS gives a better performance than when considering the worst-case scenario. Instead, the worst-case scenario might be useful for CC latency optimization.

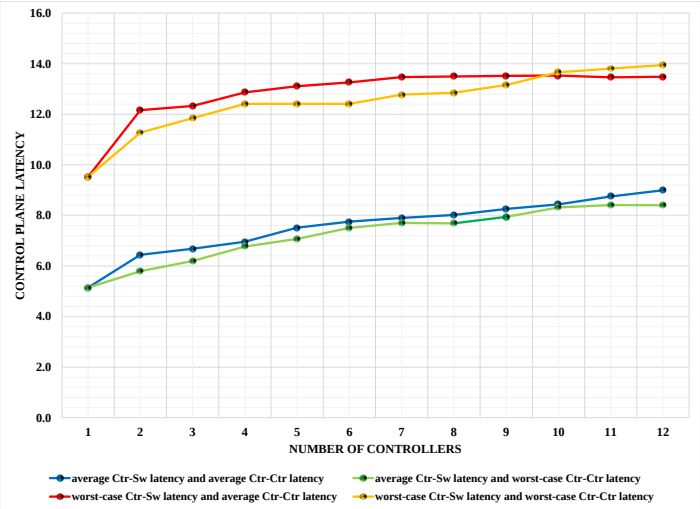

Fig. 5.13 Control plane latency in Internet2 OS3E topology

5.5 Chapter Summary

Placing multiple controllers is an essential requirement for managing large-scale networks. In this chapter, we considered the accumulated latency to solve the placement of controllers, which determines the number of required controllers and the location of the controllers to achieve a joint optimization between the controller and its associated switches, as well as to enhance inter-controller communications. We conducted experiments on Internet2 OS3E real network topology. We propose that, depending on the control plane applications, a value for lambda will be selected. Running different applications in the control plane may require different values of λ. According to the value of λ, we provide the solution to determine the optimal placement of the controllers for each case. The results show that as the number of controllers increases and the placement of controllers becomes more distributed, the controllers get closer to the switches and the latency of the CS decreases. On the other hand, if the controllers are fewer and more concentrated, then the latency of the CC decreases. Once the value of lambda is set according to the control plane application, the optimal number of controllers to balance CS and CC latencies may be found. Finally, results show that average latency optimization is better than worst-case for CS.

Chapter 6

Joint Latency and Reliability-Aware Controller Placement

6.1 Introduction

In this chapter, we first propose a new metric on the controller placement that simultaneously considers the communication latency and communication reliability both between switches and controllers, and between controllers. Reliability is considered for single-link failure (SLF). We model the problem of determining the optimal controller placement to provide low latencies in the control plane traffic. The objective of this study is to minimize the average accumulated latency, by jointly taking into account the latency between controller to switches and inter-controller, while optimizing their placement for simultaneously achieving optimal balance. The optimization problem is formulated as a Mixed Integer Linear Programming (MILP) model under the constraints of latency and reliability. We evaluated the performance of our proposed metric by using the Internet2 OS3E network topology. Different from previous work, we focus on the control traffic exchanged among controllers to synchronize their shared data structure. Results demonstrate that the proposed method is promising.

The rest of the chapter is organized as follows. Section 6.2 gives a description of related work in the CPP. We formulate the problem mathematically in Section 6.3. Section 6.4 presents our results in the control plane. Finally, the findings are summarized in Section 6.5.

6.2 Related Work

In controller placements, reliability is also a key problem in improving the network performance of the network. Yuqi Fan et al. [9] proposed a new latency metric, Latency Aware

Reliable Controller placement algorithm (LARC), for the problem that focuses on the latency between the switches and their assigned controller's communication with an single-link failure (SLF), and proposed an efficient algorithm for the problem. The objective to minimize the average accumulated latency on both primary and backup paths is jointly considered during the placement of controllers, but this approach does not consider the communication among controllers.

Yuqi Fan and Tao Ouyang [141] proposed an efficient Reliability-aware Controller Placement (RCP) algorithm against the state of the art LARC [9] and the optimized K-means network partition algorithm (OKMP) [67]. The RCP algorithm divides the network into multiple sub-networks and places a controller in each sub-network. A local search algorithm is presented to determine controller placements and switch-to-controller mapping relationship. The simulation results showed that the RCP algorithm provides better performance to reduce the backup path latency and the accumulated latency of the primary and backup paths. However, this did not address CC communication.

Singh et al. [142] focused on the RCPP and use Varna-Based Optimization (VBO) for a reliable CPP that minimizes the total average latency of the network. Results show that the proposed algorithm provides better performance compared with other optimization methods, i.e., PSO [138], Teacher Learning-Based Optimization (TLBO) [140], [143], and Jaya algorithms [144] based solution for RCPP.

Sahoo et al. [109] propose Particle Swarm Optimization (PSO) and Firefly (FFA) for solving controller placement problems in SDN-based wide area networks. The objective of their work was to minimize the latency to find the optimal number and location of the controller, taking into account the communication between the switch to controller and controller to controller. Their findings showed that FFA outperforms the PSO in terms of performance, and that the time required by both is nearly identical.

In our study, we investigated a new metric for Joint Latency and Reliability-aware Controller Placement Optimization (LRCP), which takes into account both the communication reliability and the communication latency between the controller and their associated switches as well as the controller-to-controller (simultaneously). Reliability is considered in terms of susceptibility to single-link failure (SLF).

6.3 Optimization Model

An SDN network is represented as a graph $G = (V, E)$, where switches (or nodes) set V consists of SDN-enabled network devices and edges (or links). Set E includes the communication links among devices, where the link weight indicates the distance between two nodes.

The set of switches V and edges E are defined in 6.1 and 6.2:

$$V = \{v_i | i = 1, 2, ..., n\} \tag{6.1}$$

$$E = \{link(v_i, v_j) | v_i, v_j \subset V\} \tag{6.2}$$

Where, v_i and v_j are switch nodes, link (v_i, v_j) is the link between nodes v_i and v_j and n denotes the total amount of nodes ($n = |V|$). C represents the set of controller nodes in the network topology, and E_C represents the set of paths between the controllers, which are shown in 6.3 and 6.4:

$$C = \{c_i | i = 1, 2, ..., k\} \tag{6.3}$$

$$E_C = \{link(c_i, c_j) | c_i, c_j \subset C\} \tag{6.4}$$

Where, c_i and c_j are controller nodes, link (c_i, c_j) is the link between nodes c_i and c_j, and k is the number of controllers to be deployed throughout the network ($k \leq n$). M represents the distance matrix between all nodes, and $dist_{(i,j)}$ represents the distance between nodes i and j. A node may be a simple switch node or a controller node which is placed on the switch node. For example, in Figure 5.1 there are 12 nodes from switches and 3 are controllers, thus n = 12 and k = 3. The details of the notation in this chapter are summarized in Table 6.1.

We propose a new metric called Joint Latency and Reliability-aware Controller Placement Optimization (LRCP) based on the original CPP model [6], with some additional constraints and terms for the objective function. As a first approach, a metric is proposed to find the trade-off between the controller and their associated switch latency (CS) and inter-controller latency (CC) at the same time. The goal of this metric is to minimize the accumulated latency of the control plane, mathematically formulated as described in 6.5:

$$\text{joint latency} = \lambda * \text{CS(RL)} + (1 - \lambda) * \text{CC(RL)} \tag{6.5}$$

Where CS(RL) and CC(RL) represent the CS and CC latency, respectively, taking into account the effect of a single-link failure (SLF). The objective function in the first part of the model formulation considers the latency between switches and their assigned controller (CS(RL) weighted by λ) and the second part of the function considers the controller-to-controller latency (CC(RL) weighted by (1-λ)). A detailed evaluation of this metric and its applicability has been presented [8]. The goal of the model is to determine the optimal

$G(V,E)$	Graph G, where V is a set of switches and E is a set of edges between switches
V	Set of switches or nodes in the network
E	Set of physical links or edges between switches
C	Set of controllers to be installed, where ($C \subset V$)
i, j	Switches or nodes i and j
(i, j)	Link from between nodes i and j
$dist_{(i,j)}$	Distance between nodes i and j
n	Total number of nodes in the network, where ($n = \|V\|$)
k	Total number of controllers to be installed on the network, where ($k \leq n$)
$p_{i,k}$	Link set of the primary path between switch i and controller k
$l^p_{i,k}$	Latency of the primary path between switch i and controller k
$BPL_{i,k}$	Average latency of the backup paths between switch i and controller k
$l^b_{i,k,i',j'}$	Latency of the backup path between switch–controller under the link failure (i', j')
$RL(i,k)$	Accumulated latency on both primary and backup paths between switch–controller

Table 6.1 Notations used for LRCP model

minimum accumulated latency placements and controller locations to achieve a given balance between switches to the assigned controller latencies, and inter-controllers latencies in the control plane. The latency between two nodes is measured as the distance by calculating the shortest path distance between pairs of nodes using Dijkstra's algorithm. According to this model, each controller is placed with a switch, and a single controller controls each switch.

The new LRCP metric is built as an extension of [8], including reliability for single-link failures (SLFs), following the work presented in LARC [9]. $\mathrm{RL}(i, j)$ represents the shortest distance between switch node i to controller node j including the average latency of the backup path when a single-link failure occurs. n represents the total amount of nodes and k represents the number of controllers.

Average switch to controller latency (CS)

The average latency between controllers and switches is calculated as shown in 6.6:

$$CS(RL) = \frac{\sum_{i \in V, j \in C} \mathrm{RL}_{(i,j)} x_{i,j}}{(n-k)} \tag{6.6}$$

This optimization problem is known as the minimum K-means clustering algorithm problem.

Average inter-controller latency (CC)

The average latency is shown in (6.7)

$$CC(RL) = \frac{\sum_{i \in V, j \in C} \mathrm{RL}_{(i,j)} x_{i,i} y_{j,j}}{n_{(CC)}} \tag{6.7}$$

The number of possible paths between controller-controller is dependent on the number of controllers k and is 6.8:

$$n_{(CC)} = k(k-1)/2 \tag{6.8}$$

$$\mathrm{RL} = \theta * PPL + (1-\theta) * BPL \tag{6.9}$$

We assume a single-link failure (SLF). Reliability-Latency (RL) is represented by Equation 6.9, which includes both the average latency of the primary path as well as an average of the alternative backup paths that may be used if one single-link of the primary path fails. RL is used for both CS and CC paths. The average latency of the primary paths is computed as shown in Equation (6.10).

$$PPL = \frac{\sum_{i \in V, k \in C} l^p_{i,k} \cdot x_{i,k}}{n} \tag{6.10}$$

In the case of an single-link failure (SLF) of the primary path $l^p_{i,k}$, the backup path needs to be built and updated to replace the failed primary path in the network.

$$BPL_{i,k} = \frac{\sum_{(i',j') \in p_{i,k}} l^b_{i,k,i',j'}}{|p_{i,k}|} \tag{6.11}$$

$$BPL = \frac{\sum_{i \in V, k \in C} l^b_{i,k} \cdot x_{i,k}}{n} \tag{6.12}$$

When an single-link failure (SLF) occurs (i', j') in the primary path $l^p_{i,k}$, the backup path needs to be set up to connect switch i and its assigned controller k. We apply Dijkstra's algorithm to find the shortest path which avoids the failed link to rebuild the connection between switch i and controller k. Since every link on primary path $l^p_{i,k}$ may fail, we compute the average latency of the backup path between switch i and controller k. By considering all

single-link failure (SLF) in the network, the average latency of all the backup paths in the network is computed by Equation (6.11) and Equation (6.12).

This model is formulated under the following constraints:

$$\sum_{j \in C} x_{i,j} = 1, \forall i \in V \tag{6.13}$$

$$\sum_{i \in V} y_{i,j} = 1, \forall j \in C \tag{6.14}$$

$$y_{i,j} \leq x_{i,j}, \forall i \in V, \forall j \in C \tag{6.15}$$

$$x_{i,j}, y_{i,j} \in \{0,1\} \tag{6.16}$$

These constraints ensure that: each switch is assigned to exactly one controller (6.13); each controller is located on exactly one switch (6.14); a switch i is mapped to the controller j if controller j is placed on switch i (6.15) and Constraint (6.16) guarantees that $x_{i,j}$ and $y_{i,j}$ are binary integer variables (equal to 0 or 1). Binary decision variables are described as in (6.17) and (6.18):

$$x_{i,j} = \begin{cases} 1, & \text{if switch } i \text{ is mapped to controller } j. \\ 0, & \text{otherwise.} \end{cases} \tag{6.17}$$

$$y_{i,j} = \begin{cases} 1, & \text{if controller } k \text{ is placed on switch } i. \\ 0, & \text{otherwise.} \end{cases} \tag{6.18}$$

Using RL latency, we build a topology graph where the weights of the links include an extra value representing the average backup path latency. Optimization using this topology aims to determine the best placement of a set of controllers when an single-link failure (SLF) occurs.

6.4 Performance Evaluation

We aim to minimize the total accumulated average latency, taking into consideration the latency between switches and their assigned controller as well as the latency among controllers simultaneously with a single-link failure (SLF). The real network topology used is the Internet2 OS3E network (34 nodes and 42 links) [136]. Mathematical modelling is written in Python. The optimization program uses Gurobi Optimizer to solve the MILP model [111].

In our model, we have two different types of weights, λ and θ. The value of the λ is utilized for the weights between CS and CC latencies. Depending on the considered control plane application, more weight for minimizing CS traffic or CC traffic may be required. On the other hand, the value of the θ is used for the weights of the latency of the primary path

(*PPL*) and the average latency of the backup path (*BPL*). Therefore, the controller location may vary according to both parameters λ or θ.

When $\theta = 1$ and $\lambda = 1$, the results depend on the average latency between switches and their assigned controllers only and considers the primary paths only. This case corresponds with the basic CPP and LARC optimization using the primary paths only [6], [9]. Average latency decreases when more controllers are used. As expected, we obtain the same results as CPP and LARC; that is, the same controller placements and set of switches per controller, and the same optimal controller to switch latency.

The case when $\theta = 1$ with different values of λ corresponds with the basic Joint Latency Controller Placement Optimization (LCP) [8]. Figure 6.1 depicts the comparison results of average accumulated latency and the impact of λ on latency when the number of controllers varied from 2 to 12. It is interesting to note that, as the number of controllers deployed reaches a certain value in the control plane, the average accumulated latency begins to increase. By increasing the number of controllers, the latency among controllers increases. As the value of λ grows, the average accumulated latency increases.

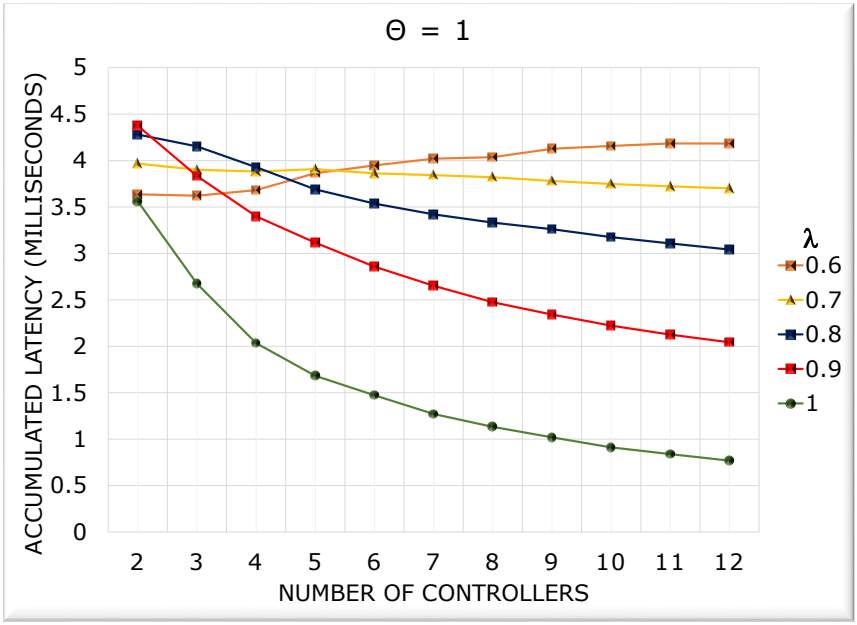

Fig. 6.1 Average accumulated latency (milliseconds) using primary path only ($\theta = 1$)

The reliability of the control plane decreases as the failure probability increases. Therefore, controller placements influence both network latency and reliability in the whole network. Figure 6.2 presents the corresponding results when $\theta = 0.9$. The accumulated

latency takes into account the average of backup paths and this implies slightly different controller placements and associations with their corresponding switches. As an example, for 5 controllers ($k = 5$) and $\lambda = 0.8$ the accumulated latency considering the primary paths is 3.69 ms (Figure 6.1) while considering the cost of a link failure with $\theta = 0.9$ the new placement gives an accumulated latency of 4.29 ms.

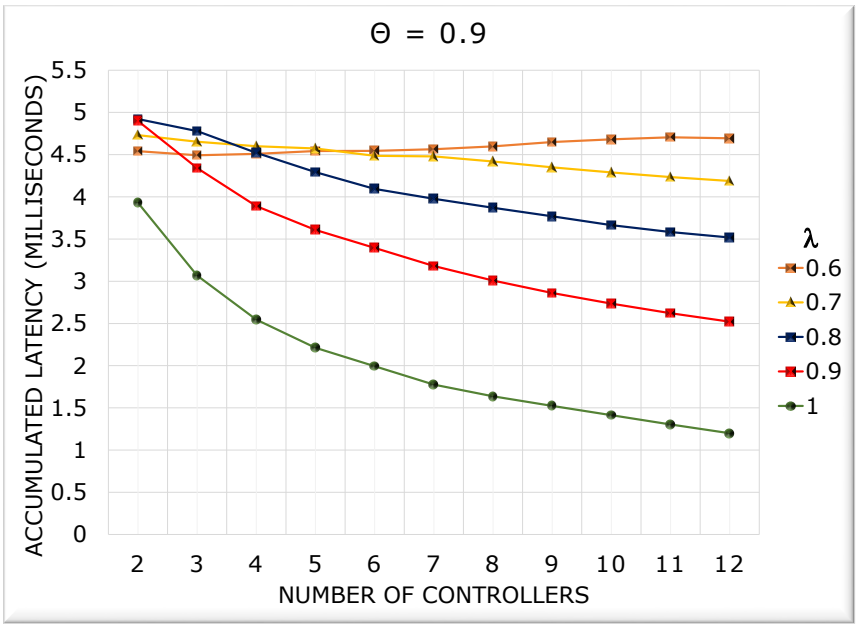

Fig. 6.2 Average accumulated latency (milliseconds)($\theta = 0.9$)

Figure 6.3 shows how different values of θ influence the placement of a set of controllers and the distribution of switches attached to each controller. For five controllers ($k = 5$), Fig. 6.3 (a) corresponds to (Fig. 6.1) when $\theta = 1$ and $\lambda = 0.8$ while Fig. 6.3 (b) corresponds to (Fig. 6.2) when $\theta = 0.9$ and $\lambda = 0.8$.

For a given probability of a link failure, the sub-optimal placement that considers this event and corresponding backup paths provides a slightly larger value of the accumulated latency respect value for the primary paths.

Figure 6.4 analyzes the effect of the number of controllers on the average accumulated latency between switch and controller ($\lambda = 1$). These results are the same as the original LARC proposal [9]. We observe that by increasing θ, the average accumulated latency decreases. Remember that $\theta = 1$ indicates that we are only using the primary paths.

Figure 6.5 shows the average accumulated latency of our proposal (LRCP) for a particular case when the number of controllers is five ($k = 5$). For $\theta = 1$, LCP [8] and LRCP are

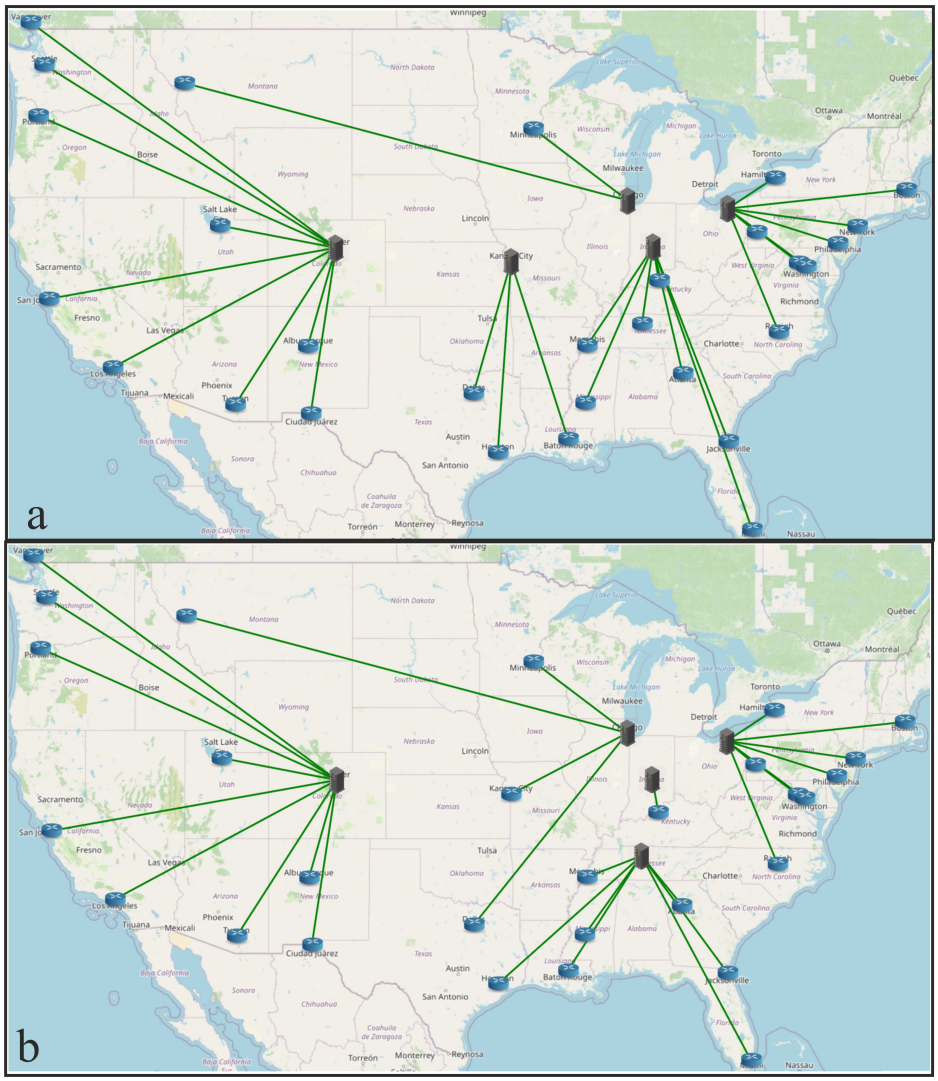

Fig. 6.3 Optimal controller location and their associated switches, (a) $\theta = 1$, (b) $\theta = 0.9$

the same. In particular, the case when $\lambda = 1$ corresponds to the previous work of CPP and LARC for $\theta = 1$. CPP places the controllers to optimize the primary path for the CS latency only without considering link failures [6]. LARC deploys the controllers intending to minimize the average latency of primary and backup paths between controllers and their associated switches (CS) when an single-link failure (SLF) occurs. However, CC latency is not considered in LARC [9]. The average latency increases while the weight of backup path latency growth (θ decreases) because the average latency of the backup path is larger than the average latency of the primary path.

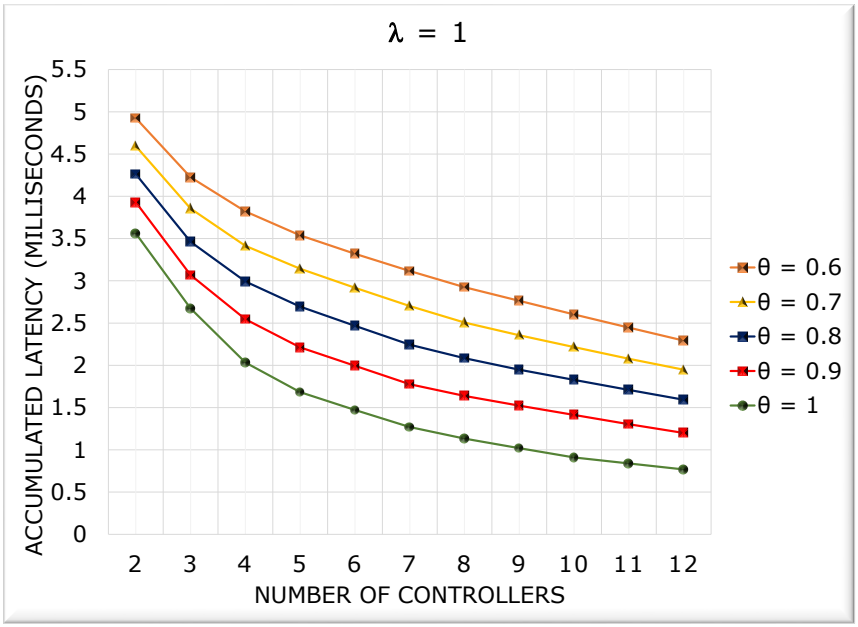

Fig. 6.4 Average accumulated latency (milliseconds) for switch-controller ($\lambda = 1$)

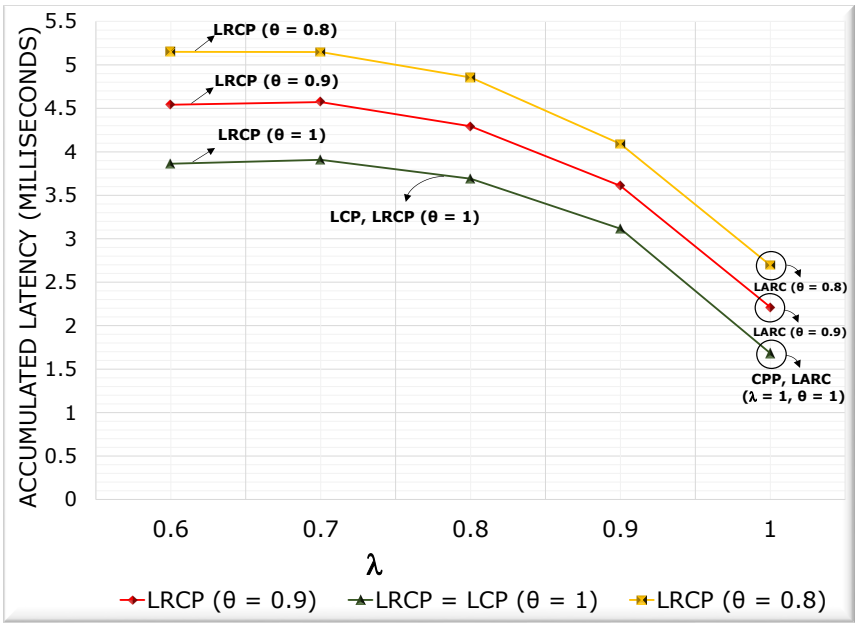

Fig. 6.5 Average accumulated latency (milliseconds) of LRCP for 5 controllers (k = 5)

6.5 Chapter Summary

Placing multiple controllers is an essential requirement to improve network performance and reliability. We tackled the original CPP by extending it and considering the communication latency and communication reliability. We conducted experiments using Internet2 OS3E network topology, so that we may compare our results with the previous works. Our proposed metric LRCP is built as an extension of the CPP and LCP models including reliability for single-link failures (SLFs), following the work presented in the LARC model. The latency among controllers increases as the number of controllers becomes larger. When the number of controllers decreases, the latency between controllers to switch traffic increases.

The two parameters of the model (λ and θ) play a key role in the deployment of the controllers. Once the control plane application is decided, the corresponding CC communication may be included in the controller placement optimization by choosing the value of λ. The optimal performance of the control plane depends both on the latency between switches and controllers (CS) and the latency among controllers (CC). On the other hand, the parameter θ allows finding better locations for the controllers considering the probability of link failures.

A greater failure probability will imply lower values of θ, so that the alternative backup paths have a larger weight in the placement decision. In future research, we plan to consider the failure of several links while solving the controller placement, and thus analyze a more practical scenario.

Chapter 7

Evaluation of Joint Controller Placement for Latency and Reliability-Aware Control Plane

7.1 Introduction

This chapter presents an evaluation of the LRCP optimization model presented in Chapter 6. LRCP provides network administrators with flexible choices to simultaneously achieve a trade-off between the switch-to-controller latency and the controller-to-controller latency, including the reliability aspect using alternative backup paths. Control Plane Latency (CPL) is used as the evaluation metric and it is defined as the sum of average switch-to-controller latency and the average inter-controller latency. For each optimal placement in the network, the control plane latency using the real latencies of the real network topology is computed. Results from the control plane latency metric show how the number and location of controllers influence the reliability of the network. In the event of a Single-Link Failure (SLF), the real CPL for LRCP placements is computed and assesses how good the LRCP placements are.

This chapter proposes the CPL metric to evaluate controller placements that simultaneously considers both latency and reliability. The structure of the remainder of this chapter is organized as follows. Section 7.2 describes the state of the art of the controller placement problem in software-defined networks. Section 7.3 presents a summary of the proposal we aim to evaluate. Section 7.4 presents the performance evaluation. Finally, Section 7.5 summarizes the main findings and the achieved results.

7.2 Related Work

Reliability is a crucial concern for controller placements. Hu et al. implement a metric known as the expected percentage of control path loss for a failed network component; this is used to characterize the reliability of SDN control networks. They present a heuristic *l-w*-greedy algorithm to evaluate the trade-off between reliability and latency. The aim of this optimization is to minimize the expected percentage of control path loss. In this case, the reliability metric is defined as the expected percentage of control path loss, where the control path loss is the number of broken control paths due to network failures [130], [125]. LARC was proposed to minimize the average latency between all the switches and their corresponding controllers when an single-link failure (SLF) occurs [9]. The latency of each path includes the primary path latency and an average of the corresponding possible backup paths under an single-link failure (SLF) condition. Yuqi Fan et al. proposed an efficient Reliability-aware Controller Placement (RCP) algorithm for multiple controller placements that splits the network into multiple subnetworks and places a controller in each subnetwork [141]. A local search algorithm determines the controller locations and maps the relationship between switches and controllers. The simulation results show that the proposed RCP algorithm can effectively reduce the latency of the primary and backup paths.

Varna-Based Optimization (VBO) offers reliable CPP with minimized total average latency [142]. Empirical investigations revealed that the VBO algorithm gives better performance than other efficient heuristic algorithms, such as PSO [138], TLBO [140], Jaya algorithm [144] solutions for RCPP. The LRCP optimization model for controller placement considers both latencies between switches to their controllers and the latency between controllers at the same time (i.e., joint latency) [145]. It considers the reliability impact of a single failure, referred to as “reliability-aware”.

7.3 Description of the LRCP optimization

LRCP is an extension of CPP models [6], presenting the basic controller placement problem, and LCP [8], which defines the joint latency as an optimization function. Furthermore, it includes the reliability aspect following the LARC model for an single-link failure (SLF) [9]. LRCP is defined to reduce the accumulated latency by integrating the two sub-objectives of reliability and joint latency. Latency is approximated by the distance between the nodes. The optimization of LRCP is formulated as a MILP under both latency and reliability constraints. It must be noted that the LRCP optimization is done offline and thence its complexity is not a drawback. A summary of the original proposal is presented in this section.

RL is defined as the weighted sum of the primary and the backup path latencies (7.1). The primary path latency (PPL) between two nodes is based on the shortest path routing algorithm. The backup path latency (BPL) is the average latency of all the possible alternative paths when an single-link failure (SLF) in the primary path occurs. To do the computation of the average backup path latency, a single-link along the primary path is removed from the network and a new shortest path between the nodes is calculated. The RL between a pair of nodes is defined as:

$$\mathrm{RL} = \theta * \mathrm{PPL} + (1-\theta) * \mathrm{BPL} \tag{7.1}$$

Parameter θ assigns the weight to the real latency (PPL) between two nodes and the additional average latency in case of a failure (BPL).

The CS latency is the average latency between a controller and all its associated switches. The average latency of the inter-controller communications (CC) is the average latency of all possible paths between pairs of controllers. The joint latency is defined as the weighted sum of the average controller to switch latency (CS) and the average inter-controller latency (CC). This is referred to as joint latency (7.2), where λ is the weight between the two latencies. Both CS and CC are computed using the RL in order to include the reliability factor. The joint latency is the optimization function and it is defined as:

$$\text{joint latency} = \lambda * \mathrm{CS(RL)} + (1-\lambda) * \mathrm{CC(RL)} \tag{7.2}$$

Both parameters of the LRCP optimization (λ and θ) play a key role in controller placements. The optimal placement may vary depending on either λ or θ. A particular placement is defined by the location of the controllers and the set of switches associated with each controller. Once the control plane application is decided, λ is used to find a trade-off for the performance of the control plane application. For instance, when λ is close to one it means that the control plane application needs very fast communication between switches and their controller, and the inter-controller latency is not a key factor for the performance. The optimal performance of the control plane depends on a balance between the latencies CS and CC.

Parameter θ allows finding better controller placements, considering the probability of link failures. A greater probability of link failure may need lower values of θ, so that the alternative backup paths have a larger weight in the placement decision. The case when θ = 1 is considered as the PPL only by ignoring BPL. We may see θ as a parameter to find preventive placements that may be affected less than the original one by single-link failure (SLF). It is interesting and useful to make an assessment of the control plane performance

of these preventive placements. The major functions and details of the LRCP optimization model and the results can be found in [145].

7.4 Performance Evaluation

LRCP and performance evaluation was undertaken on the Internet2 OS3E network topology. OS3E contains 34 nodes and 42 edges. The goal of the evaluation is to assess the goodness of the optimal placements provided by LRCP in a real deployment. For each value of λ and θ, LRCP gives the optimal placement (location of the controllers and the set of switches associated with each controller). When a particular placement is deployed in the network, the important metric is the overall control plane latency. Thus, the proposed metric for the evaluation is CPL, defined as the sum of the average CS and CC latencies. For each optimal placement found, CPL is computed as:

$$CPL = \mathrm{CS} + \mathrm{CC} \tag{7.3}$$

CPL is computed using the actual latencies of the real network topology. The PPL, when no failures occur, is taken as the reference value.

In order to include a degree of reliability, in case a link failure event occurs, the preventive placement considers this event and the average BPL is added in the optimization. This provides a slightly larger value of the CPL value with respect to the average of the primary path latency only (PPT). For the evaluation, two cases are differentiated. The first one does not include any link failure, and it intends to assess the real CPL for the LRCP preventive placements which include a weight for the BPL (i.e., θ less than 1). The second case computes the real CPL when an single-link failure (SLF) occurs and assesses how good are the LRCP preventive placements. Intuitively, when an single-link failure (SLF) occurs, an LRCP preventive placement obtained for $\theta < 1$ should behave better than the one obtained for PPL only ($\theta = 1$). The goal is to quantify their performance.

7.4.1 Control Plane Latency Evaluation Without Link Failure

Figure 7.1 illustrates the impact of the number of controllers (k) on the CPL. The case in which $\theta = 1$ with different values of λ corresponds to the average PPL. For $\theta < 1$, the placement considers the average BPL and implies slightly different controller locations and associated switches. As expected, it provides a higher control plane latency, because it does not correspond with the optimal placement when no failures occur. The case with $\theta = 0.5$, shown in Figure 7.1, indicates that for 5 controllers ($k = 5$) and $\lambda = 0.5$, the average control

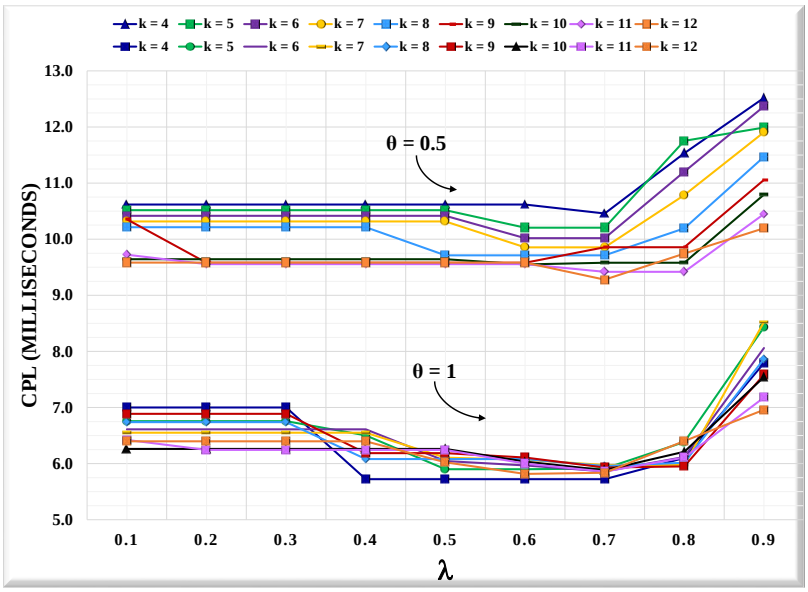

Fig. 7.1 CPL without a single-link failure (SLF)

plane latency for $\theta = 1$ is 5.9 ms. Considering the cost of an single-link failure (SLF) with $\theta = 0.5$, the preventive placement gives an average control plane latency of 10.5 ms; this is an increase of about 78%. Unless there is a high probability of link failures, higher values for θ should be used; otherwise, the sought protection is counterproductive.

Useful values for θ are those close to 1. Figure 7.2 shows the relationship between the control plane latency and the number of controllers by varying the weights of θ when $\lambda = 0.5$. It is interesting to note that the control plane latency grows linearly by increasing the weight of backup path latency (decreasing values of θ), since the average BPL provides a larger value of the control plane latency with respect to the value for the average primary path latency (PPT). When 5 controllers are deployed ($k = 5$) and $\lambda = 0.5$, the control plane latency decreases from 8.1 ms to 5.9 ms when θ ranges between 0.8 and 1. This means an increase of 38% in the control plane latency with respect to the primary paths if there is no failure. For $\lambda = 0.6$ (not shown in the figure), the control plane latency is 7.7 ms with respect to 5.9 ms; that is an increment of 31%. It can be seen from Figure 7.2 that giving more weight to the backup path (i.e., lower values of θ) produces preventive placements with an increase in the control plane latency. In conclusion, values of less than $\theta = 0.8$ are not useful, because the CPL increases more than 30% if there are no failures.

In order to compare LRCP evaluation with previous proposals and as a validation of the study, Figure 7.3 presents the average control plane latency for the case of 5 controllers ($k =$

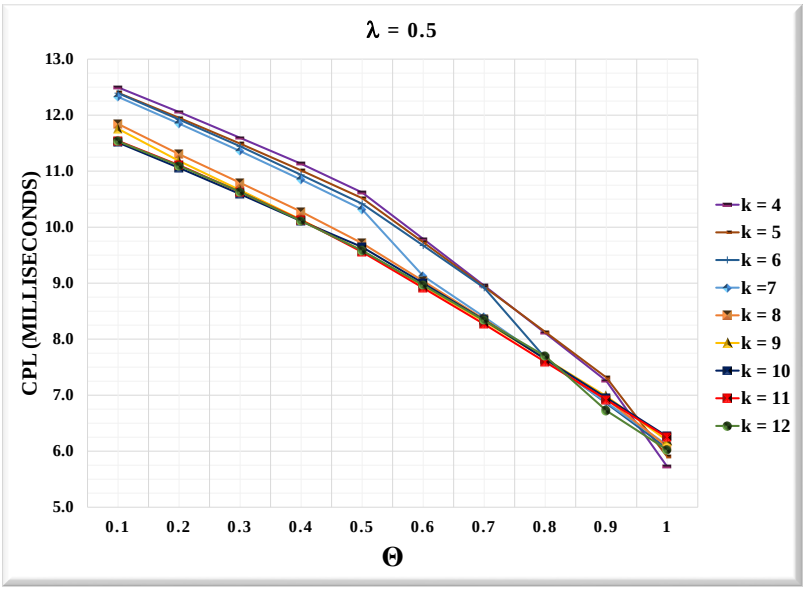

Fig. 7.2 CPL and $\lambda = 0.5$

5). For $\theta = 1$, using different values of λ, the result corresponds with the basic joint latency controller placement optimization (LCP) [8] and LRCP [145].

The LCP optimization model minimizes the joint latency and the number of network controllers providing placements with a balance between CS and CC latencies. However, the LCP optimization model does not consider the probability of a link failure. In particular, the case when $\lambda = 1$ corresponds to the original optimization of the CPP, and for $\theta = 1$ it corresponds with the LARC.

The CPP places the controllers to minimize the latency between switches and controllers only without considering the failure scenario [6].

The LARC algorithm aims to minimize the accumulated latency by integrating the two sub-objectives: the average of the primary path latency, and the backup path latency into one objective (between switches and controllers) when an single-link failure (SLF) occurs [9].

As a last example of the cost associated with using preventive placements, when 5 controllers are placed and the value of λ is set to 0.5, it can be observed in Figure 7.3 that the control plane latency with $\theta = 1$ is 5.9 ms, whereas it is 7.3 ms in case of $\theta = 0.9$. This means an increase of 24%; depending on the probability of link failures, this extra latency might be tolerated.

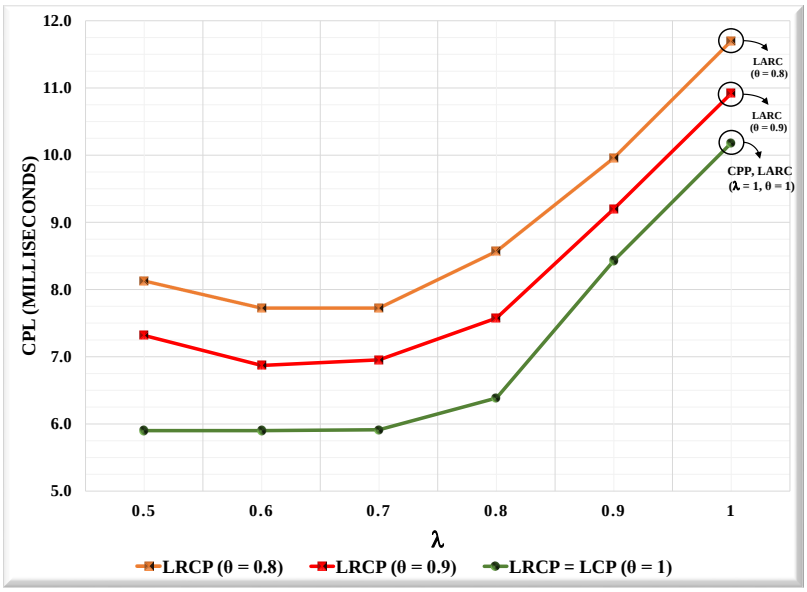

Fig. 7.3 CPL for 5 controllers: LRCP, LARC, LCP, and CPP

7.4.2 Control Plane Latency Evaluation for Single-Link Failure (SLF)

In this section, the CPL is computed for LRCP placements considering the effect of single-link failure. A link failure may affect CS paths, CC paths, both, for 1 controller or for several controllers at the same time. To compute the average CPL, for each placement (location of the controllers and the set of assigned switches to each controller), one link i is removed and CPL (i) is computed. The final value of the CPL metric is the average of CPL (i) for all the links being removed one at a time.

Figure 7.4 presents the control plane latency with an single-link failure (SLF) for 5 controllers ($k = 5$). Comparing the results shown in Figures 7.3 and 7.4, for the placements including the primary latency (PPL) only ($\theta = 1$), the average control plane latency for $\lambda = 0.6$ is 5.9 ms (see Figure 7.3), and is 6.4 ms in the case of a single-link failure (see Figure 7.4). This is an increase of about 10%. For $\theta = 0.8$ and $\lambda = 0.6$ the CPL is 7.7 ms when there are no failures (Figure 7.3) and 6.4 ms with a single-link failure (SLF). This means a relative decrease of about 14%.

This confirms that for low link failure probabilities, using values of $0.8 < \theta < 1$, provides reliable placements with a reasonable increase of about 10% in CPL, respecting the reference values (when no links fail). However, this increase is compensated when a link fails with a relative decrease of about 14%.

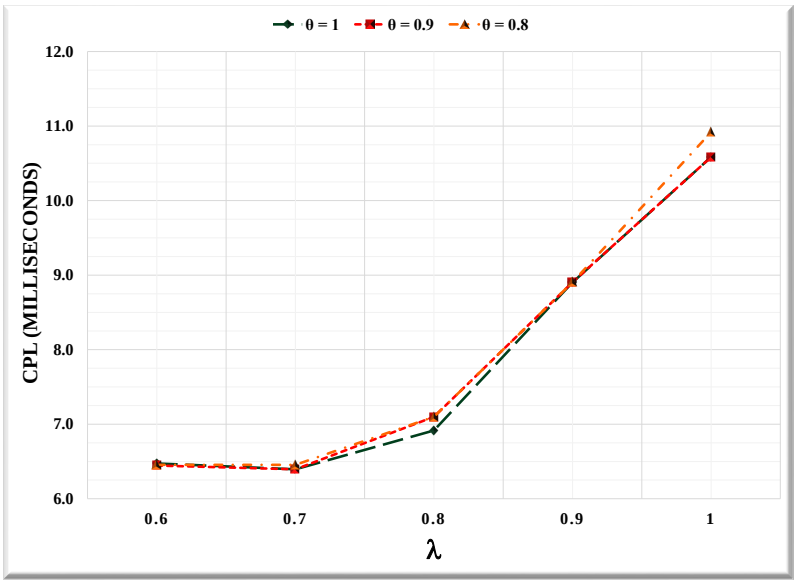

Fig. 7.4 CPL with single-link failure (SLF), k =5

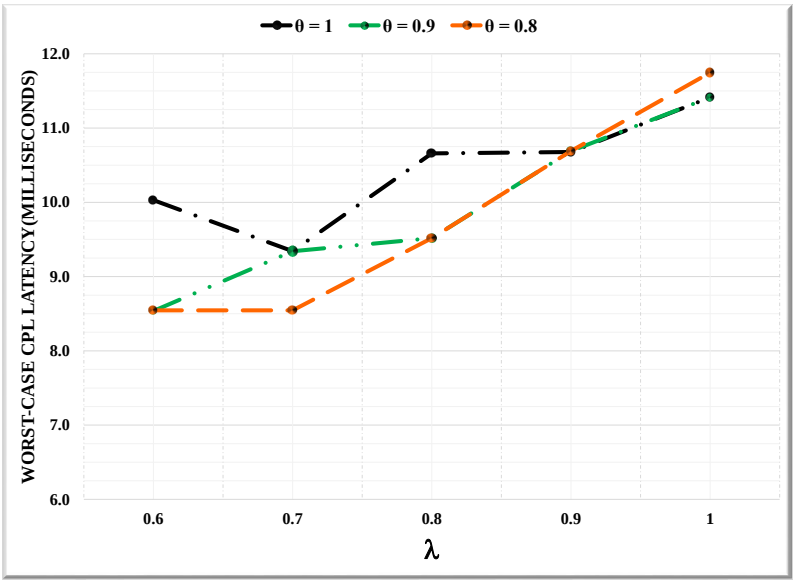

Fig. 7.5 Worst-case CPL latency with single-link failure (SLF), k = 5

As mentioned before, in the case of a single-link failure (SLF), the computed CPL is an average latency. It is then relevant to investigate the worst-case when a link fails. Figure 7.5 presents the worst-case control plane latency when an single-link failure (SLF) occurs.

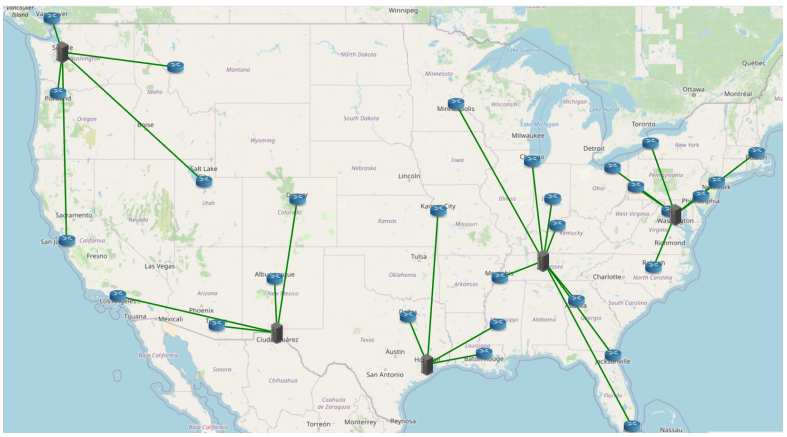

Fig. 7.6 Optimal controller locations and sets of switches associated with each controller (θ, $\lambda = 1$)

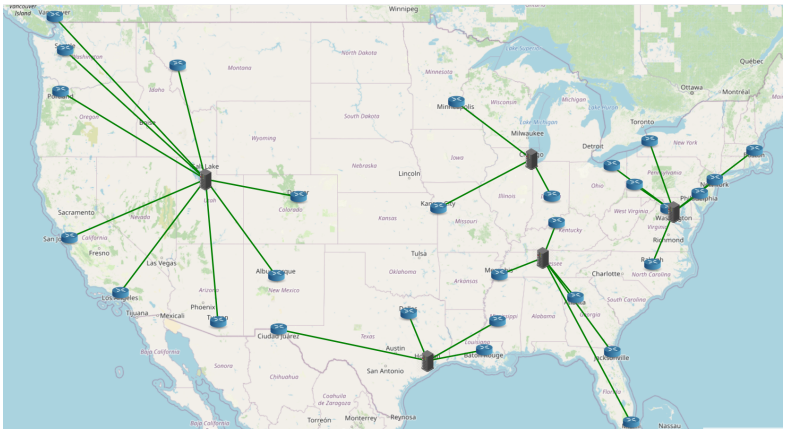

Fig. 7.7 Optimal controller locations and sets of switches associated with each controller (θ, $\lambda = 0.9$)

For instance, for $k = 5$ when θ ranges from 0.8 to 1 and, the control plane latency for $\lambda = 0.8$ ranges from 9.5 ms to 10.6 ms, and for $\lambda = 0.6$ ranges from 8.5 ms to 10.0 ms. In other words, for $\lambda = 0.8$ the worst-case CPL increases by 12%, and for $\lambda = 0.6$ it increases by 17%.

It is worth mentioning that two nodes have exactly one link in the Internet2 OS3E network: Miami and Vancouver. These two switches could get disconnected from their associated controller if the corresponding link fails and there is no possible backup path.

Figures 7.6 and 7.7 show how different values of θ and λ influence the placement of controllers that manage the control plane, and the distribution of all switches assigned to each controller.

Tables 7.1, 7.2 and 7.3 shows the details of further results. The same color means identical placements (controller locations and sets of switches associated with each controller), and thus the same CPL. The tables contain results for values of θ between 0.8 and 1 because, from the previous evaluation results, these are the useful values for a real deployment.

For a given placement, fixed by the combination of λ and θ , CPL is the average control plane latency during a normal operation (without failures), while CPL (sf) is the average control plane latency when an single-link failure (SLF) occurs. The CPL-CPL(sf) is the relative increase of CPL (sf) with respect to CPL. As can be observed in Table 7.3, preventive placements with $\theta = 0.8$ implies slightly higher values of CPL and smaller differences for CPL (sf). The table also includes the worst-case value of CPL and its variance for each placement. It is up to the network manager to choose the appropriate values of the parameters and use a preventive placement instead of the one without considering backup paths.

	$\theta = 1$				
Weights of λ	**$\lambda = 0.6$**	**$\lambda = 0.7$**	**$\lambda = 0.8$**	**$\lambda = 0.9$**	**$\lambda = 1$**
CPL(sf)	**6.47**	**6.39**	**6.92**	**8.90**	**10.58**
CPL	**5.90**	**5.91**	**6.39**	**8.43**	**10.18**
CPL-CPL(sf)	**0.57**	**0.48**	**0.53**	**0.47**	**0.40**
CPL-CPL(sf)	**10%**	**8%**	**8%**	**6%**	**4%**
Min CPL(sf)	**5.90**	**5.91**	**6.39**	**8.43**	**10.18**
wc CPL(sf)	**10.03**	**9.34**	**10.66**	**10.68**	**11.41**
ratio wc CPL / min CPL	**70%**	**58%**	**67%**	**27%**	**12%**
Var CPL	**1.03**	**0.80**	**0.91**	**0.53**	**0.34**

Table 7.1 Comparison of the control plane latency with and without single-link failure (SLF) for 5 controllers when $\lambda = 1$

	$\theta = 0.9$				
Weights of λ	**$\lambda = 0.6$**	**$\lambda = 0.7$**	**$\lambda = 0.8$**	**$\lambda = 0.9$**	**$\lambda = 1$**
CPL(sf)	**6.44**	**6.39**	**7.10**	**8.90**	**10.58**
CPL	**6.02**	**5.91**	**6.58**	**8.44**	**10.18**
CPL-CPL(sf)	**0.43**	**0.48**	**0.51**	**0.46**	**0.40**
CPL-CPL(sf)	**7%**	**8%**	**8%**	**5%**	**4%**
Min CPL(sf)	**6.02**	**5.91**	**6.58**	**8.44**	**10.18**
wc CPL(sf)	**8.54**	**9.34**	**9.52**	**10.69**	**11.41**
ratio wc CPL / min CPL	**42%**	**58%**	**45%**	**27%**	**12%**
Var CPL	**0.65**	**0.80**	**0.81**	**0.53**	**0.34**

Table 7.2 Comparison of the control plane latency with and without single-link failure (SLF) for 5 controllers when $\lambda = 0.9$

	θ = 0.8				
Weights of λ	**λ = 0.6**	**λ = 0.7**	**λ = 0.8**	**λ = 0.9**	**λ = 1**
CPL(sf)	**6.45**	**6.45**	**7.10**	**8.90**	**10.92**
CPL	**6.02**	**6.02**	**6.58**	**8.44**	**10.51**
CPL-CPL(sf)	**0.07**	**0.07**	**0.08**	**0.05**	**0.04**
CPL-CPL(sf)	**43%**	**43%**	**51%**	**46%**	**40%**
Min CPL(sf)	**6.02**	**6.02**	**6.58**	**8.44**	**10.51**
wc CPL(sf)	**8.55**	**8.55**	**9.52**	**10.69**	**11.75**
ratio wc CPL / min CPL	**42%**	**42%**	**45%**	**27%**	**12%**
Var CPL	**0.65**	**0.65**	**0.81**	**0.53**	**0.34**

Table 7.3 Comparison of the control plane latency with and without single-link failure (SLF) for 5 controllers when $\lambda = 0.8$

7.5 Chapter Summary

In this study, the average CPL metric has been used to evaluate the LRCP optimization model. CPL is defined as the sum of the average switch to controller latency (CS) and the average inter-controller- latency (CC). LRCP provides optimal placements, simultaneously considering balance between CS and CC latencies, and a reliability tradeoff with preventive placements in case an single-link failure (SLF) occurs. Parameters λ and θ are used, respectively, for each goal. Preventive placements consider the corresponding backup paths.

The main contribution of the chapter is the assessment of the goodness of LRCP preventive placements in a real deployment. In order to quantify the evaluation, the CPL metric is used. From the evaluation results presented in LRCP [145], the reference values selected for the evaluation were 5 controllers ($k = 5$) and values of λ from 0.6 to 1. From the results obtained, we may conclude that preventive placements for values of $\theta < 0.8$ are not advisable, because they introduce too much extra latency in the control plane in a normal operation without failures. On the other hand, placements with $\theta >= 0.8$ give a good tradeoff between the added latency while in normal operation and when an single-link failure (SLF) occurs.

In summary, for low link failure probabilities using values $0.8 < \theta < 1$ provide reliable preventive placements with a reasonable increase of the CPL in respect to the reference values (when no links fail), and this increase is compensated when a link fails with a relative decrease of CPL.

Chapter 8

Conclusions and Future Work

The Controller Placement Problem (CPP) is an important consideration during the design phase of the control plan using distributed controller architectures. In recent years, CPP has received significant attention, with many studies conducted on this topic exploring various aspects of the problem. The CPP decides the optimal locations to place the controllers and minimizes the number of controllers. In this study, we tackled optimizing the placement of controllers that simultaneously consider two technical aspects: communication latency and communication reliability. Our goal is to identify latency or delay between each switch and its controller, as well as between controller pairs. Although we consider the reliability factor in the event of a single-link failure (SLF). The optimal placement of the controllers must consider not only the latencies between the switches and their assigned controllers but also the latencies among controllers. Each controller in the control plane is required to communicate with the other controllers, introducing some latency in order to synchronize their internal data structures, which define the network state and provide applications with a centralized view of the network state. To the best of our knowledge, few studies have taken into account the control traffic exchanged between controllers in the control plane. To solve CPP, the majority of prior research has applied to random network topologies, but we evaluated the performance using real-world internet topologies. Consequently, we believe that our investigation provides a reliable methodology for designing networks that support the control plane in large-scale networks.

In Chapter 4, we provided a rule for the DC placements to make it easy to determine how many controllers are required in a given topology. Therefore, we noticed that the number of DCs for each cluster was similarly based on the metrics of K-means and K-center, respectively. In this case study on NRENs Western European, the optimal placement of MCs revealed that a single MC is adequate to handle the entire network and obtain the best performance in the control plane. The MC is the main controller that oversees the

management of DCs in various nations, whereas the DC oversees switches in the same country or nearby countries.

The results validate the methodology and demonstrate its applicability and feasibility on large networks and different domains. A useful use case may be the deployment of hierarchical levels of controllers for the enforcement of very precise routing policies through different domains.

The overall contribution is the methodology presented to define a multi-level control plane as an iterative optimization problem. For this particular topology, the results show that a third level is not needed, but other large topologies with different constraints may need a third level in the control plane. The study revealed that our method is very powerful for searching for all possible controller placements when analyzing the results for large networks. These requirements can help to achieve low service latency, short synchronization latency and better link utilization in the deployment of some controller instances on a given network.

In Chapter 5, the results show that as the number of controller's increases and their placement becomes more distributed, the controllers get closer to the switches and CS latency decreases. Conversely, if the controllers are fewer and more concentrated, CC latency decreases. Once the λ value is determined based on the control plane application, the optimal number of controllers to balance CS and CC latencies can be determined. Finally, the results showed that the CS optimization of average latency is better than the worst-case scenario for CS.

In Chapter 6, once the control plane application is decided, the corresponding CC communication may be included in the controller placement optimization by choosing the value of λ. The optimal performance of the control plane depends on both the latency between switches and controllers (CS) and the latency among controllers (CC). On the other hand, the parameter θ allows finding better locations for the controllers considering the probability of link failures. A greater failure probability will imply lower values of θ, so that the alternative backup paths have a larger weight in the placement decision.

In Chapter 7, we conclude that preventive placements for values of $\theta < 0.8$ are not advisable, because of the introduction of too much extra latency in the control plane for standard operations (without failure). On the other hand, placements with $\theta >= 0.8$ provide a good tradeoff between the added latency during normal operation and when an SLF occurs. In conclusion, for low-link failure probabilities, using values $0.8 < \theta < 1$ provides reliable preventive placements with a reasonable increase in CPL with respect to the reference values (when no links fail), and this increase is compensated when a link fails with a relative decrease in CPL.

In the future, we intend to extend our work to address the reliability and capacity of controller constraints while deciding the placement of the controller in the hierarchical control plane architecture. We plan to expand our investigation to include larger topologies (i.e. Full European NREN emulation model (1157 routers)). In our work, only the controllers are placed vertically, and each oversees a set of switches in their domains in the control plane. Our models may be extended to encompass the hierarchical control plane of multiple controllers, which is required to run coordination and consensus algorithms to maintain the controllers synchronized with one another. In this thesis, reliability is considered for single-link failure. In future research, we intend to take into account the failure of multiple links along with placement problems and thus analyze a more practical scenario.

References

[1] Jie Lu, Zhen Zhang, Tao Hu, Peng Yi, and Julong Lan. A survey of controller placement problem in software-defined networking. *IEEE Access*, 7(May):24290–24307, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2893283.

[2] Ashutosh Kumar Singh and Shashank Srivastava. A survey and classification of controller placement problem in SDN. *International Journal of Network Management*, 28(3), 2018. ISSN 10991190. doi: 10.1002/nem.2018.

[3] Ahmed Abdelaziz, Ang Tan Fong, A. Gani, Usman Garba, Suleman Khan, Adnan Akhunzada, Hamid Talebian, and Kim-Kwang Raymond Choo. Distributed controller clustering in software defined networks. *PLoS ONE*, 12, 2017.

[4] Volkan Yazici, M. O. Sunay, and A. Ercan. Controlling a software-defined network via distributed controllers. *ArXiv*, abs/1401.7651, 2014.

[5] Hailan Kuang, Yiwen Qiu, Ruifang Li, and Xinhua Liu. A hierarchical K-means algorithm for controller placement in SDN-Based WAN architecture. *Proceedings - 10th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2018*, 2018-January:263–267, 2018. doi: 10.1109/ICMTMA.2018.00070.

[6] Brandon Heller, Rob Sherwood, and Nick Mckeown. The controller placement problem. *Computer Communication Review*, 42(4):473–478, 2012. ISSN 01464833. doi: 10.1145/2377677.2377767.

[7] Jianxin Liao, Haifeng Sun, Jingyu Wang, Qi Qi, Kai Li, and Tonghong Li. Density cluster based approach for controller placement problem in large-scale software defined networkings. *Computer Networks*, 112:24–35, 2017. ISSN 13891286. doi: 10.1016/j.comnet.2016.10.014.

[8] Kurdman Abdulrahman Rasol Rasol and Jordi Domingo-Pascual. Joint placement latency optimization of the control plane. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, 2020. doi: 10.1109/ISNCC49221.2020.9297271.

[9] Yuqi Fan, Yongfeng Xia, Weifa Liang, and Xiaomin Zhang. Latency-Aware Reliable Controller Placements in SDNs. In Qianbin Chen, Weixiao Meng, and Liqiang Zhao, editors, *Communications and Networking*, pages 152–162, Cham, 2018. Springer International Publishing. ISBN 978-3-319-66628-0.

[10] Fellow Ieee, Christian Esteve Rothenberg, Member Ieee, Siamak Azodolmolky, Senior Member Ieee, Steve Uhlig, and Member Ieee. Software-Defined Networking : A Comprehensive Survey. 103(1), 2015.

[11] Bo Yi, Xingwei Wang, Keqin Li, Sajal k. Das, and Min Huang. A comprehensive survey of Network Function Virtualization. *Computer Networks*, 133:212–262, 2018. ISSN 13891286. doi: 10.1016/j.comnet.2018.01.021.

[12] Tom Alexander, Ilker Demirkol, Amitabh Mishra, and Alberto Perotti. Mobile Communications and Networks. *IEEE Communications Magazine*, 58(9):73, 2020. ISSN 15581896. doi: 10.1109/MCOM.2020.9214392.

[13] Qazi Kamal Ud Din Arshad, Ahsan Ullah Kashif, and Ijaz Mansoor Quershi. A Review on the Evolution of Cellular Technologies. *Proceedings of 2019 16th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2019*, pages 989–993, 2019. doi: 10.1109/IBCAST.2019.8667173.

[14] Tara Ali-Yahiya. *Understanding LTE and its Performance*. 2011. ISBN 9781441964564. doi: 10.1007/978-1-4419-6457-1.

[15] Obaid Ur-Rehman and Natasa Zivic. *Wireless communications*. 2018. ISBN 9780521837163. doi: 10.1007/978-3-319-78942-2_2.

[16] Yongpeng Wu, Fuhui Zhou, Zan Li, Shunqing Zhang, Zheng Chu, and Wolfgang H. Gerstacker. *Green Communication and Networking*, volume 2018. 2018. ISBN 9780072967753. doi: 10.1155/2018/1921353.

[17] Majid Irfan Baba, Naira Nafees, Insha Manzoor, Kamran Aijaz Naik, and Suhaib Ahmed. Evolution of Mobile Wireless Communication Systems from 1G to 5G : A Comparative Analysis. *NCRACIT) International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2018 IJSRCSEIT*, 1 (4):1–08, 2018. ISSN 2456-3307. URL http://ijsrcseit.com/paper/CSEIT411801.pdf%0Awww.ijsrcseit.com.

[18] Timo Halonen, Javier Romero, and Juan Melero. *GSM, GPRS and EDGE Performance*. 2003. ISBN 0470866942. doi: 10.1002/0470866969.

[19] Rajesh Yadav. Challenges and Evolution of Next generation Wireless Communication. *Lecture Notes in Engineering and Computer Science*, 2228:619–623, 2017. ISSN 20780958.

[20] Jochen Schiller. Mobile Communications (2nd Edition). page 492, 2003.

[21] Polynia V Kharbuli and Amina Sultana. A Comparative Study on the Generations of Mobile Wireless Telephony : 1G - 5G. *Journal of Emerging Technologies and Innovative Research.*, 5(September 2018):326–332, 2018.

[22] Mohsen Attaran. The impact of 5G on the evolution of intelligent automation and industry digitization. *Journal of Ambient Intelligence and Humanized Computing*, (0123456789), 2021. ISSN 18685145. doi: 10.1007/s12652-020-02521-x. URL https://doi.org/10.1007/s12652-020-02521-x.

[23] Gerhard P. Fettweis. The tactile internet: Applications and challenges. *IEEE Vehicular Technology Magazine*, 9(1):64–70, 2014. ISSN 15566072. doi: 10.1109/MVT.2013.2295069.

[24] Lokesh B. Bhajantri and S. Gangadharaiah. A comprehensive survey on resource management in internet of things. *Journal of Telecommunications and Information Technology*, 2020(4):27–43, 2020. ISSN 18998852. doi: 10.26636/JTIT.2020.145220.

[25] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, 2014. ISSN 23274662. doi: 10.1109/JIOT.2014.2306328.

[26] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347–2376, 2015. ISSN 1553877X. doi: 10.1109/COMST.2015.2444095.

[27] Ekram Hossain and Monowar Hasan. 5G cellular: Key enabling technologies and research challenges. *IEEE Instrumentation and Measurement Magazine*, 18(3):11–21, 2015. ISSN 10946969. doi: 10.1109/MIM.2015.7108393.

[28] Jeffrey G. Andrews, Stefano Buzzi, Wan Choi, Stephen V. Hanly, Angel Lozano, Anthony C.K. Soong, and Jianzhong Charlie Zhang. What will 5G be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, 2014. ISSN 07338716. doi: 10.1109/JSAC.2014.2328098.

[29] Stefano Buzzi, I. Chih-Lin, Thierry E. Klein, H. Vincent Poor, Chenyang Yang, and Alessio Zappone. A survey of energy-efficient techniques for 5G networks and challenges ahead. *IEEE Journal on Selected Areas in Communications*, 34(4):697–709, 2016. ISSN 07338716. doi: 10.1109/JSAC.2016.2550338.

[30] Ian F. Akyildiz, Shuai Nie, Shih Chun Lin, and Manoj Chandrasekaran. 5G roadmap: 10 key enabling technologies. *Computer Networks*, 106:17–48, 2016. ISSN 13891286. doi: 10.1016/j.comnet.2016.06.010.

[31] 5G PPP Architecture Working Group. View on 5G Architecture. *Version 3.0, June 2019*, (June):21–470, 2019. URL https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf.

[32] Hamid Farhady, Hyunyong Lee, and Akihiro Nakao. Software-Defined Networking: A survey. *Computer Networks*, 81:79–95, 2015. ISSN 13891286. doi: 10.1016/j.comnet.2015.02.014. URL http://dx.doi.org/10.1016/j.comnet.2015.02.014.

[33] Bruno Astuto A. Nunes, Marc Mendonca, Xuan Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, 16(3):1617–1634, 2014. ISSN 1553877X. doi: 10.1109/SURV.2014.012214.00180.

[34] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. A Survey on Software-Defined Networking. *IEEE Communications Surveys and Tutorials*, 17(1):27–51, 2015. ISSN 1553877X. doi: 10.1109/COMST.2014.2330903.

[35] Murat Karakus and Arjan Durresi. A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN). *Computer Networks*, 112:279–293, 2017. ISSN 13891286. doi: 10.1016/j.comnet.2016.11.017.

[36] Jahan R. Gupta D. Inter-sdn controller communication: Using border gateway protocol. *White Paper by Tata Consultancy Services (TCS)*, 2014.

[37] McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., and Turner J. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69 – 74, April 2008. doi: 10.1145/1355734.1355746.

[38] Juliver De Jesus Gil Herrera and Juan Felipe Botero Vega. Network Functions Virtualization: A Survey. *IEEE Latin America Transactions*, 14(2):983–997, 2016. ISSN 15480992. doi: 10.1109/TLA.2016.7437249.

[39] ETSI GS NFV 003 V1.2.1:. Network functions virtualisation (nfv); terminology for main concepts in nfv, 2014. *(ETSI Industry Specification Group (ISG) NFV)*, .

[40] ETSI GS NFV 002 V1.2.1:. Network functions virtualisation (nfv); architectural framework, 2014a. *(ETSI Industry Specification Group (ISG) NFV)*, .

[41] ETSI GS NFV 001 V1.2.1:. Network functions virtualisation (nfv); management and orchestration, 2014b. *(ETSI Industry Specification Group (ISG) NFV)*, .

[42] Yong Li and Min Chen. Software-defined network function virtualization: A survey. *IEEE Access*, 3:2542–2553, 2015. ISSN 21693536. doi: 10.1109/ACCESS.2015.2499271.

[43] Diego Kreutz, Fernando M.V. Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015. ISSN 15582256. doi: 10.1109/JPROC.2014.2371999.

[44] Abdelrahman Abuarqoub. A review of the control plane scalability approaches in software defined networking. *Future Internet*, 12(3), 2020. ISSN 19995903. doi: 10.3390/fi12030049.

[45] Yury Andrea Jim. Scalability and Robustness of the control plane in Software-Defined Networking (SDN):. (May):1–190, 2016.

[46] Haochi Liang, Peilin Hong, and Wei Zhou. SRSO: A scalable routing scheme for large-scale open flow networks. *Proceedings - 2014 IEEE International Conference on Computer and Information Technology, CIT 2014*, pages 636–641, 2014. doi: 10.1109/CIT.2014.69.

[47] Manar Jammal, Taranpreet Singh, Abdallah Shami, and Yiming Li. Software-Defined Networking : State of the Art and Research Challenges. pages 1–24.

[48] Alireza Shirmarz and Ali Ghaffari. Taxonomy of controller placement problem (CPP) optimization in Software Defined Network (SDN): a survey. *Journal of Ambient Intelligence and Humanized Computing*, (0123456789), 2021. ISSN 18685145. doi: 10.1007/s12652-020-02754-w. URL https://doi.org/10.1007/s12652-020-02754-w.

[49] Tamal Das, Vignesh Sridharan, and Mohan Gurusamy. A Survey on Controller Placement in SDN. *IEEE Communications Surveys and Tutorials*, 22(1):472–503, 2020. ISSN 1553877X. doi: 10.1109/COMST.2019.2935453.

[50] Si Kee Yoon, Zia Khalib, N. Yaakob, and A. Amir. Controller Placement Algorithms in Software Defined Network - A Review of Trends and Challenges. *MATEC Web of Conferences*, 140, 2017. ISSN 2261236X. doi: 10.1051/matecconf/201714001014.

[51] Bala Prakasa Rao Killi and Seela Veerabhadreswara Rao. Controller placement in software defined networks: A Comprehensive survey. *Computer Networks*, 163, 2019. ISSN 13891286. doi: 10.1016/j.comnet.2019.106883.

[52] Abdunasser Alowa. Scalable Reliable Controller Placement in Software Defined Networking Concordia University. 2020.

[53] Zohaib Latif, Kashif Sharif, Fan Li, Md Monjurul Karim, Sujit Biswas, and Yu Wang. A comprehensive survey of interface protocols for software defined networks. *Journal of Network and Computer Applications*, 156:102563, 2020. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2020.102563. URL https://www.sciencedirect.com/science/article/pii/S1084804520300370.

[54] Yonghong Fu, Jun Bi, Kai Gao, Ze Chen, Jianping Wu, and Bin Hao. Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 569–576, 2014. doi: 10.1109/ICNP.2014.91.

[55] Fetia Bannour, Sami Souihi, and Abdelhamid Mellouk. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Communications Surveys and Tutorials*, 20 (1):333–354, 2018. ISSN 1553877X. doi: 10.1109/COMST.2017.2782482.

[56] Tao Hu, Zehua Guo, Thar Baker, and Julong Lan. Multi-controller Based Software-Defined Networking : A Survey. 2017.

[57] Tianzhu Zhang, Paolo Giaccone, Andrea Bianco, and Samuele De Domenico. The role of the inter-controller consensus in the placement of distributed SDN controllers. *Computer Communications*, 113:1–13, 2017. ISSN 01403664. doi: 10.1016/j.comcom.2017.09.007.

[58] Manish Paliwal, Deepti Shrimankar, and Omprakash Tembhurne. Controllers in SDN: A review report. *IEEE Access*, 6:36256–36270, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2846236.

[59] Kurdman Abdulrahman Rasol and Jordi Domingo-Pascual "Multi-level Hierarchical Controller Placement in Software Defined Networking. In: Ghita B., Shiaeles S. (eds) Selected Papers from the 12th International Networking Conference. INC 2020. Lecture Notes in Networks and Systems, vol 180. Springer, Cham. https://doi.org/10.1007/978-3-030-64758-210 .

[60] Tianzhu Zhang and Nokia Bell Labs. Master Degree in Computer and Communication Networks Master of Science Thesis Distributed Controllers in Software Defined Networks Supervisors. (October 2014), 2019. doi: 10.13140/RG.2.2.25786.64963.

[61] Othmane Blial, Mouad Ben Mamoun, and Redouane Benaini. An Overview on SDN Architectures with Multiple Controllers. *Journal of Computer Networks and Communications*, 2016, 2016. ISSN 2090715X. doi: 10.1155/2016/9396525.

[62] Mathis Obadia, Mathieu Bouet, Jean Louis Rougier, and Luigi Iannone. A greedy approach for minimizing SDN control overhead. *1st IEEE Conference on Network Softwarization: Software-Defined Infrastructures for Networks, Clouds, IoT and Services, NETSOFT 2015*, pages 15–19, 2015. doi: 10.1109/NETSOFT.2015.7116135.

[63] Zhiyang Su and Mounir Hamdi. MDCP: Measurement-aware distributed controller placement for software defined networks. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, 2016-January:380–387, 2016. ISSN 15219097. doi: 10.1109/ICPADS.2015.55.

[64] Abdunasser Alowa and Thomas Fevens. Combined Degree-Based with Independent Dominating Set Approach for Controller Placement Problem in Software Defined Networks. *Proceedings of the 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops, ICIN 2019*, (Icin):269–276, 2019. doi: 10.1109/ICIN.2019.8685897.

[65] Deze Zeng, Chao Teng, Lin Gu, Hong Yao, and Qingzhong Liang. Flow setup time aware minimum cost switch-controller association in Software-Defined Networks. *Proceedings of the 11th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QSHINE 2015*, pages 259–264, 2015. doi: 10.4108/eai.19-8-2015.2260893.

[66] Guodong Wang, Yanxiao Zhao, Jun Huang, and Yulei Wu. An Effective Approach to Controller Placement in Software Defined Wide Area Networks. *IEEE Transactions on Network and Service Management*, 15(1):344–355, 2018. ISSN 19324537. doi: 10.1109/TNSM.2017.2785660.

[67] Guodong Wang, Yanxiao Zhao, Jun Huang, Qiang Duan, and Jun Li. A K-means-based network partition algorithm for controller placement in software defined network. *2016 IEEE International Conference on Communications, ICC 2016*, pages 16–21, 2016. doi: 10.1109/ICC.2016.7511441.

[68] Bala Prakasa Rao Killi and Seela Veerabhadreswara Rao. On Placement of Hypervisors and Controllers in Virtualized Software Defined Network. *IEEE Transactions on Network and Service Management*, 15(2):840–853, 2018. ISSN 19324537. doi: 10.1109/TNSM.2018.2823341.

[69] Mu He, Arsany Basta, Andreas Blenk, and Wolfgang Kellerer. Modeling flow setup time for controller placement in SDN: Evaluation for dynamic flows. *IEEE International Conference on Communications*, 2017. ISSN 15503607. doi: 10.1109/ICC.2017.7996654.

[70] Qinghong Zhong, Ying Wang, Wenjing Li, and Xuesong Qiu. A min-cover based controller placement approach to build reliable control network in SDN. *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, (Noms):481–487, 2016. doi: 10.1109/NOMS.2016.7502847.

[71] Yannan Hu, Wendong Wang, Xiangyang Gong, Xirong Que, and Shiduan Cheng. On reliability-optimized controller placement for Software-Defined Networks. *China Communications*, 11(2):38–54, 2014. ISSN 16735447. doi: 10.1109/CC.2014.6821736.

[72] Jiang Liu, Juan Liu, and Renchao Xie. Reliability-based controller placement algorithm in software defined networking. *Computer Science and Information Systems*, 13 (2):547–560, 2016. ISSN 24061018. doi: 10.2298/CSIS160225014L.

[73] Ying Zhang, Neda Beheshti, and Mallik Tatipamula. On resilience of split-architecture networks. *GLOBECOM - IEEE Global Telecommunications Conference*, 2011. doi: 10.1109/GLOCOM.2011.6134496.

[74] Bala Prakasa Rao Killi and Seela Veerabhadreswara Rao. Optimal Model for Failure Foresight Capacitated Controller Placement in Software-Defined Networks. *IEEE Communications Letters*, 20(6):1108–1111, 2016. ISSN 10897798. doi: 10.1109/LCOMM.2016.2550026.

[75] Bala Prakasa Rao Killi and Seela Veerabhadreswara Rao. Capacitated Next Controller Placement in Software Defined Networks. *IEEE Transactions on Network and Service Management*, 14(3):514–527, 2017. ISSN 19324537. doi: 10.1109/TNSM.2017.2720699.

[76] Nancy Perrot and Thomas Reynaud. Optimal placement of controllers in a resilient SDN architecture. *Proceedings of the 2016 12th International Conference on the Design of Reliable Communication Networks, DRCN 2016*, (Drcn):145–151, 2016. doi: 10.1109/DRCN.2016.7470849.

[77] Petra Vizarreta, Carmen Mas MacHuca, and Wolfgang Kellerer. Controller placement strategies for a resilient SDN control plane. *Proceedings of 2016 8th International Workshop on Resilient Networks Design and Modeling, RNDM 2016*, pages 253–259, 2016. doi: 10.1109/RNDM.2016.7608295.

[78] Neda Beheshti and Ying Zhang. Fast failover for control traffic in Software-defined Networks. *GLOBECOM - IEEE Global Telecommunications Conference*, pages 2665–2670, 2012. doi: 10.1109/GLOCOM.2012.6503519.

[79] Maryam Tanha, Dawood Sajjadi, and Jianping Pan. Enduring node failures through resilient controller placement for software defined networks. *2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings*, 2016. doi: 10.1109/GLOCOM.2016.7841786.

[80] Diogo M.F. Mattos, Otto Carlos M.B. Duarte, and Guy Pujolle. A resilient distributed controller for software defined networking. *2016 IEEE International Conference on Communications, ICC 2016*, 2016. doi: 10.1109/ICC.2016.7511032.

[81] He Li, Robson Eduardo De Grande, and Azzedine Boukerche. An efficient CPP solution for resilience-oriented SDN controller deployment. *Proceedings - 2017 IEEE 31st International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2017*, pages 540–549, 2017. doi: 10.1109/IPDPSW.2017.161.

[82] Lucas F. Muller, Rodrigo R. Oliveira, Marcelo C. Luizelli, Luciano P. Gaspary, and Marinho P. Barcellos. Survivor: An enhanced controller placement strategy for improving SDN survivability. *2014 IEEE Global Communications Conference, GLOBECOM 2014*, pages 1909–1915, 2014. doi: 10.1109/GLOCOM.2014.7037087.

[83] Sheng Guo, Shu Yang, Qi Li, and Yong Jiang. Towards Controller Placement for robust Software-Defined Networks. *2015 IEEE 34th International Performance Computing and Communications Conference, IPCCC 2015*, 2016. doi: 10.1109/PCCC.2015.7410301.

[84] S. Sedef Savas, Massimo Tornatore, M. Farhan Habib, Pulak Chowdhury, and Biswanath Mukherjee. Disaster-resilient control plane design and mapping in software-defined networks. *IEEE International Conference on High Performance Switching and Routing, HPSR*, 2016-June:0–5, 2016. ISSN 23255609. doi: 10.1109/HPSR.2015.7483086.

[85] Bala Prakasa Rao Killi and Seela Veerabhadreswara Rao. Link failure aware capacitated controller placement in software defined networks. *International Conference on Information Networking*, 2018-January:292–297, 2018. ISSN 19767684. doi: 10.1109/ICOIN.2018.8343128.

[86] Bala Prakasa Rao Killi and Seela Veerabhadreswara Rao. Towards improving resilience of controller placement with minimum backup capacity in software defined networks. *Computer Networks*, 149:102–114, 2019. ISSN 13891286. doi: 10.1016/j.comnet.2018.11.027. URL https://doi.org/10.1016/j.comnet.2018.11.027.

[87] F. Aykut Özsoy and Mustafa Ç Pinar. An exact algorithm for the capacitated vertex p-center problem. *Computers and Operations Research*, 33(5):1420–1436, 2006. ISSN 03050548. doi: 10.1016/j.cor.2004.09.035.

[88] Guang Yao, Jun Bi, Yuliang Li, and Luyi Guo. On the capacitated controller placement problem in software defined networks. *IEEE Communications Letters*, 18(8):1339–1342, 2014. ISSN 10897798. doi: 10.1109/LCOMM.2014.2332341.

[89] Hemant Kumar Rath, Vishvesh Revoori, S. M. Nadaf, and Anantha Simha. Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game. *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, WoWMoM 2014*, 2014. doi: 10.1109/WoWMoM.2014.6918987.

[90] Hidenobu Aoki and Norihiko Shinomiya. Network Partitioning Problem for Effective Management of Multi-domain SDN Networks. *International Journal on Advances in Networks and Services*, 8(3-4):62–77, 2015. ISSN 1942-2644. URL https://thinkmind.org/index.php?view=article{&}articleid=netser{_}v8{_}n34{_}2015{_}5.

[91] Genya Ishigaki and Norihiko Shinomiya. Controller placement algorithm to alleviate burdens on communication nodes. *2016 International Conference on Computing, Networking and Communications, ICNC 2016*, 2016. doi: 10.1109/ICCNC.2016.7440618.

[92] Maryam Tanha, Dawood Sajjadi, Rukhsana Ruby, and Jianping Pan. Capacity-Aware and Delay-Guaranteed Resilient Controller Placement for Software-Defined WANs. *IEEE Transactions on Network and Service Management*, 15(3):991–1005, 2018. ISSN 19324537. doi: 10.1109/TNSM.2018.2829661.

[93] Junichi Nagano and Norihiko Shinomiya. Efficient information sharing among distributed controllers of OpenFlow network with bi-connectivity. *2015 International Conference on Computing, Networking and Communications, ICNC 2015*, pages 320–324, 2015. doi: 10.1109/ICCNC.2015.7069362.

[94] Hidenobu Aoki, Junichi Nagano, and Norihiko Shinomiya. Network Partitioning Problem to Reduce Shared Information in OpenFlow Networks with Multiple Controllers. (c):250–255, 2015.

[95] J. Nagano and N. Shinomiya. Controller placement problem to enhance performance in multi-domain SDN networks. *Computing, Networking and Communications (ICNC), International Conference on. IEEE, 2015*, page 320–324, 2015.

[96] Md Tanvir Ishtaique Ul Huque, Guillaume Jourjon, and Vincent Gramoli. Revisiting the controller placement problem. *Proceedings - Conference on Local Computer Networks, LCN*, 26-29-October-2015:450–453, 2015. doi: 10.1109/LCN.2015.7366350.

[97] Qingxiang Lin and Dong Zhang. Traffic-Aware compatible controller deployment. *Proceedings of the 2015 10th International Conference on Communications and Networking in China, CHINACOM 2015*, pages 847–852, 2016. doi: 10.1109/CHINACOM.2015.7498055.

[98] Bangzhou Liu, Binqiang Wang, and Xiaoqiang Xi. Heuristics for SDN controller deployment using community detection algorithm. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 0:253–258, 2016. ISSN 23270594. doi: 10.1109/ICSESS.2016.7883061.

[99] Kshira Sagar Sahoo, Bibhudatta Sahoo, Ratnakar Dash, and Mayank Tiwary. Solving multi-controller placement problem in software defined network. *Proceedings - 2016 15th International Conference on Information Technology, ICIT 2016*, pages 188–192, 2017. doi: 10.1109/ICIT.2016.60.

[100] Md Faizul Bari, Arup Raton Roy, Shihabur Rahman Chowdhury, Qi Zhang, Mohamed Faten Zhani, Reaz Ahmed, and Raouf Boutaba. Dynamic controller provisioning in software defined networks. *2013 9th International Conference on Network and Service Management, CNSM 2013 and its three collocated Workshops - ICQT 2013, SVM 2013 and SETM 2013*, pages 18–25, 2013. doi: 10.1109/CNSM.2013.6727805.

[101] Mostafa Khorramizadeh and Vahid Ahmadi. Capacity and load-aware software-defined network controller placement in heterogeneous environments. *Computer Communications*, 129(August):226–247, 2018. ISSN 1873703X. doi: 10.1016/j.comcom.2018.07.037. URL https://doi.org/10.1016/j.comcom.2018.07.037.

[102] Vahid Ahmadi and Mostafa Khorramizadeh. An adaptive heuristic for multi-objective controller placement in software-defined networks. *Computers and Electrical Engineering*, 66:204–228, 2018. ISSN 00457906. doi: 10.1016/j.compeleceng.2017.12.043. URL https://doi.org/10.1016/j.compeleceng.2017.12.043.

[103] Yury Jimenez, Juan Antonio Cordero, and Cristina Cervello-Pastor. Measuring robustness of SDN control layers. *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, pages 774–777, 2015. doi: 10.1109/INM.2015.7140373.

[104] Fetia Bannour, Sami Souihi, and Abdelhamid Mellouk. Scalability and reliability aware SDN controller placement strategies. *2017 13th International Conference on Network and Service Management, CNSM 2017*, 2018-January:1–4, 2017. doi: 10.23919/CNSM.2017.8255989.

[105] Yang Fu, Fan Ning, Xin Li, Bingli Guo, Yu Zhou, Jie Zhang, and Shanguo Huang. A novel multi-controller placement scheme against single controller failure in software defined optical networks. *Optics InfoBase Conference Papers*, (c):13–15, 2014. ISSN 2162108X. doi: 10.1364/ACPC.2016.AF2A.148.

[106] Eugen Borcoci, Tudor Ambarus, and Marius Vochin. Multi-criteria based Optimization of Placement for Software Defined Networking Controllers and Forwarding Nodes. *The Fifteenth International Conference on Networks ICN 2016*, (February), 2016. URL https://www.thinkmind.org/index.php?view=instance{&}instance=ICN+2016{%}0Ahttps://www.thinkmind.org/download.php?articleid=icn{_}2016{_}5{_}30{_}37021.

[107] Bang Zhang, Xingwei Wang, Lianbo Ma, and Min Huang. Optimal controller placement problem in internet-oriented software defined network. *Proceedings - 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2016*, pages 481–488, 2017. doi: 10.1109/CyberC.2016.98.

[108] Ahmad Jalili, Manijeh Keshtgari, Reza Akbari, and Reza Javidan. Multi criteria analysis of Controller Placement Problem in Software Defined Networks. *Computer Communications*, 133(August 2018):115–128, 2019. ISSN 1873703X. doi: 10.1016/j.comcom.2018.08.003. URL https://doi.org/10.1016/j.comcom.2018.08.003.

[109] Kshira Sagar Sahoo, Anamay Sarkar, Sambit Kumar Mishra, Bibhudatta Sahoo, Deepak Puthal, Mohammad S. Obaidat, and Balqies Sadun. Metaheuristic solutions for solving controller placement problem in SDN-based WAN architecture. *ICETE 2017 - Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, 1(September):15–23, 2017. doi: 10.5220/0006483200150023.

[110] Guodong Wang, Yanxiao Zhao, Jun Huang, and Wei Wang. The Controller Placement Problem in Software Defined Networking: A Survey. *IEEE Network*, 31(5):21–27, 2017. ISSN 1558156X. doi: 10.1109/MNET.2017.1600182.

[111] *Gurobi Optimization. (2015). Gurobi Optimizer Reference Manual. [Online]. Available: http://www.gurobi.com.*

[112] *IBM ILOG. Cplex optimizer. http://www 01.ibm.com/software/commerce/optimization/cplex-optimizer; 2012.*

[113] Franciscus X.A. Wibowo, Mark A. Gregory, Khandakar Ahmed, and Karina M. Gomez. Multi-domain Software Defined Networking: Research status and challenges. *Journal of Network and Computer Applications*, 87:32–45, 2017. ISSN 10958592. doi: 10.1016/j.jnca.2017.03.004.

[114] Peng Xiao, Wenyu Qu, Heng Qi, Zhiyang Li, and Yujie Xu. The sdn controller placement problem for wan. In *2014 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 220–224, 2014. doi: 10.1109/ICCChina.2014.7008275.

[115] Bala Prakasa Rao Killi, Ellore Akhil Reddy, and Seela Veerabhadreswara Rao. Cooperative game theory based network partitioning for controller placement in sdn. In *2018 10th International Conference on Communication Systems Networks (COMSNETS)*, pages 105–112, 2018. doi: 10.1109/COMSNETS.2018.8328186.

[116] David Hock, Steffen Gebert, Matthias Hartmann, Thomas Zinner, and Phuoc Tran-Gia. POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks. *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, 2014. doi: 10.1109/NOMS.2014.6838275.

[117] Bassey Isong, Reorapetse Ramoliti Samuel Molose, Adnan M. Abu-Mahfouz, and Nosipho Dladlu. Comprehensive review of SDN controller placement strategies. *IEEE Access*, 8:170070–170092, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3023974.

[118] Stanislav Lange, Steffen Gebert, Thomas Zinner, Phuoc Tran-Gia, David Hock, Michael Jarschel, and Marco Hoffmann. Heuristic approaches to the controller placement problem in large scale sdn networks. *IEEE Transactions on Network and Service Management*, 12(1):4–17, 2015. doi: 10.1109/TNSM.2015.2402432.

[119] Lingxia Liao and Victor C. M. Leung. Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in sdns. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–6, 2017. doi: 10.1109/NFV-SDN.2017.8169836.

[120] Kshira Sagar Sahoo, Sampa Sahoo, Anamay Sarkar, Bibhudatta Sahoo, and Ratnakar Dash. On the placement of controllers for designing a wide area software defined networks. In *TENCON 2017 - 2017 IEEE Region 10 Conference*, pages 3123–3128, 2017. doi: 10.1109/TENCON.2017.8228398.

[121] Adlen Ksentini, Miloud Bagaa, Tarik Taleb, and Ilangko Balasingham. On using bargaining game for optimal placement of sdn controllers. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016. doi: 10.1109/ICC.2016.7511136.

[122] Genya Ishigaki, Riti Gour, Ashkan Yousefpour, Norihiko Shinomiya, and Jason P. Jue. Cluster leader election problem for distributed controller placement in sdn. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, 2017. doi: 10.1109/GLOCOM.2017.8254748.

[123] Shaoteng Liu, Rebecca Steinert, and Dejan Kostic. Flexible distributed control plane deployment. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7, 2018. doi: 10.1109/NOMS.2018.8406150.

[124] Guang Yao, Jun Bi, and Luyi Guo. On the cascading failures of multi-controllers in software defined networks. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–2, 2013. doi: 10.1109/ICNP.2013.6733624.

[125] X. Gong X. Que Y. Hu, W. Wendong and C. Shiduan. Reliability aware controller placement for software-defined networks. *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, 2013*, pages 672–675, 2013.

[126] Shadi Moazzeni, Mohammad Reza Khayyambashi, Naser Movahhedinia, and Franco Callegati. On reliability improvement of Software-Defined Networks. *Computer Networks*, 133:195–211, 2018. ISSN 13891286. doi: 10.1016/j.comnet.2018.01.023. URL https://doi.org/10.1016/j.comnet.2018.01.023.

[127] Vignesh Sridharan, Mohan Gurusamy, and Tram Truong-Huu. On Multiple Controller Mapping in Software Defined Networks with Resilience Constraints. *IEEE Communications Letters*, 21(8):1763–1766, 2017. ISSN 10897798. doi: 10.1109/LCOMM.2017.2696006.

[128] He Li, Peng Li, Song Guo, and Amiya Nayak. Byzantine-resilient secure software-defined networks with multiple controllers in cloud. *IEEE Transactions on Cloud Computing*, 2(4):436–447, 2014. ISSN 21687161. doi: 10.1109/TCC.2014.2355227.

[129] Francisco J. Ros and Pedro M. Ruiz. On reliable controller placements in Software-Defined Networks. *Computer Communications*, 77:41–51, 2016. ISSN 01403664. doi: 10.1016/j.comcom.2015.09.008. URL http://dx.doi.org/10.1016/j.comcom.2015.09.008.

[130] Yannan Hu, Wendong Wang, Xiangyang Gong, Xirong Que, and Shiduan Cheng. On reliability-optimized controller placement for Software-Defined Networks. *China Communications*, 11(2):38–54, 2014. ISSN 16735447. doi: 10.1109/CC.2014.6821736.

[131] Yury Jiménez, Cristina Cervelló-Pastor, and Aurelio J. García. On the controller placement for designing a distributed SDN control layer. *2014 IFIP Networking Conference, IFIP Networking 2014*, 2014. doi: 10.1109/IFIPNetworking.2014.6857117.

[132] Maryam Tanha, Dawood Sajjadi, Rukhsana Ruby, and Jianping Pan. Capacity-Aware and Delay-Guaranteed Resilient Controller Placement for Software-Defined WANs. *IEEE Transactions on Network and Service Management*, 15(3):991–1005, 2018. ISSN 19324537. doi: 10.1109/TNSM.2018.2829661.

[133] C. G. Bell, A. N. Habermann, J. McCredie, R. Rutledge, and W. Wulf. *Computer networks*, volume 3. 1970. ISBN 9780132856201. doi: 10.1109/C-M.1970.216702.

[134] Guodong Wang, Yanxiao Zhao, Jun Huang, and Yulei Wu. An Effective Approach to Controller Placement in Software Defined Wide Area Networks. *IEEE Transactions on Network and Service Management*, 15(1):344–355, 2018. ISSN 19324537. doi: 10.1109/TNSM.2017.2785660.

[135] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 21–29, 2001. ISSN 07349025. doi: 10.1145/380752.380755.

[136] F. Yeung. Internet 2: scaling up the backbone for R D. *IEEE Internet Computing*, 1 (2):36–37, 1997. ISSN 0084-6597.

[137] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011. ISSN 07338716. doi: 10.1109/JSAC.2011.111002.

[138] C. Gao, Hua Wang, Fangjin Zhu, Linbo Zhai, and Shanwen Yi. A particle swarm optimization algorithm for controller placement problem in software defined network. In *ICA3PP*, 2015.

[139] Shuai Liu, Hua Wang, Shanwen Yi, and Fangjin Zhu. Ncpso: A solution of the controller placement problem in software defined networks. pages 213–225, 2015.

[140] A. Singh, N. Kumar, and Shashank Srivastava. Pso and tlbo based reliable placement of controllers in sdn. *International Journal of Computer Network and Information Security*, 11:36–42, 2019.

[141] Yuqi Fan and Tao Ouyang. Reliability-aware controller placements in software defined networks. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 2133–2140, 2019. doi: 10.1109/HPCC/SmartCity/DSS.2019.00295.

[142] Ashutosh Kumar Singh, Saurabh Maurya, Naveen Kumar, and Shashank Srivastava. Heuristic approaches for the reliable SDN controller placement problem. *Transactions on Emerging Telecommunications Technologies*, 31(2):e3761, 2020. doi: https://doi.org/10.1002/ett.3761. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3761.

[143] R Venkata Rao. *Teaching-Learning-Based Optimization Algorithm*, pages 9–39. Springer International Publishing, Cham, 2016. ISBN 978-3-319-22732-0. doi: 10.1007/978-3-319-22732-0_2. URL https://doi.org/10.1007/978-3-319-22732-0_2.

[144] Rao RV. Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int J Ind Eng Comput*, 7(1):19–34, 2016.

[145] Kurdman Abdulrahman Rasol Rasol and Jordi Domingo-Pascual. Joint latency and reliability-aware controller placement. In *2021 International Conference on Information Networking (ICOIN)*, pages 197–202, 2021. doi: 10.1109/ICOIN50884.2021.9333864.

Appendix A

List of publications

- Kurdman Abdulrahman Rasol and Jordi Domingo-Pascual “Multi-level Hierarchical Controller Placement in Software Defined Networking. In: Ghita B., Shiaeles S. (eds) Selected Papers from the 12th International Networking Conference. INC 2020. Lecture Notes in Networks and Systems, vol 180. Springer, Cham. https://doi.org/10.1007/978-3-030-64758-210
- Kurdman Abdulrahman Rasol and Jordi Domingo-Pascual, "Joint Placement Latency Optimization of the Control Plane," International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, 2020, pp. 1-6, doi: 10.1109/ISNCC49221.2020.9297271.
- Kurdman Abdulrahman Rasol and Jordi Domingo-Pascual, "Joint Latency and Reliability-Aware Controller Placement," International Conference on Information Networking (ICOIN), 2021, pp. 197-202, doi:10.1109/ICOIN50884.2021.9333864.
- Kurdman Abdulrahman Rasol and Jordi Domingo-Pascual, "Evaluation of Joint Controller Placement for Latency and Reliability-Aware Control Plane," International Conference on Software Defined Systems (SDS), Gandia, Spain. December 6-9, 2021 (to be published).