



UNIVERSIDADE DO ALGARVE

*Controller Placement in Software Defined Networking*

Mohammad Ashrafi Habibabadi

Master Thesis in Computer Science Engineering

Work done under the supervision of: Prof. Dra. Noélia Correia

2018

---

# Statement of Originality

---

## Controller Placement in Software Defined Networking

**Statement of authorship:** The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text. The material has not been submitted, either in whole or in part, for a degree at this or any other university.

**Candidate:**

---

(Mohammad Ashrafi Habibabadi)

Copyright ©Mohammad Ashrafi Habibabadi. A Universidade do Algarve tem o direito, perpétuo e sem limites geográficos, de arquivar e publicitar este trabalho através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, de o divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.



N E T W O R K I N G

Work done at Research Center of Electronics Optoelectronics and  
Telecommunications (CEOT)

---

# Abstract

---

This work focuses on the placement of controllers in software defined networking architectures. The goal is to optimize the latency besides having reliability and scalability in mind. Two mathematical models are proposed, the former determines the optimal controller location in single mapping scenarios and the latter determines the optimal location in multiple mapping scenarios. A scalability factor is introduced to equally decrease load among controllers, increasing the load to capacity gap at controllers in any failure scenario. The results show that the model finds the optimal location while taking redundancy and scalability into consideration.

Keywords: SDN, Controller Placement, Networks, Optimization

---

# Resumo

---

Este trabalho incide sobre a colocação de controladores em redes definidas por software. O objetivo é otimizar a latência, tendo em consideração a proteção em cenários de falha e a escalabilidade. São propostos dois modelos matemáticos: o primeiro determina a localização do controlador em cenários de mapeamento único (single mapping), e o último determina a localização ideal em cenários de mapeamento múltiplo. É introduzido um fator de escalabilidade para reduzir de igual forma, a carga nos controladores, havendo uma maior diferença entre a carga nos controladores e a sua capacidades, isto para qualquer cenário de falha. Os resultados mostram que o modelo consegue encontrar a localização ideal tomando em consideração a proteção em cenários de falha e a escalabilidade.

Termos chave: SDN, Colocação de controladores, Rede, Otimização

---

# Acknowledgements

---

I would like to thank Prof. Noélia for her guidance through all preparation, planning and development of this thesis. Without her help it would be very difficult to finish this journey. It was a great opportunity to work under her supervision.

I would like to thank Farog Al-Tam for his assistance through all steps required to organize the whole work. His method to solve problems is totally instructive.

---

# Contents

---

|   |            |
|---|------------|
| <b>Statement of Originality</b>                         | <b>i</b>   |
| <b>Abstract</b>   | <b>iii</b> |
| <b>Resumo</b>   | <b>iv</b>  |
| <b>Acknowledgements</b>                                 | <b>v</b>   |
| <b>Nomenclature</b>                                     | <b>xi</b>  |
| <b>1 Introduction</b>                                   | <b>1</b>   |
| 1.1 SDN . . . . .                                       | 2          |
| 1.2 SDN Layers . . . . .                                | 2          |
| 1.3 SDN Architecture . . . . .                          | 2          |
| 1.4 SDN Functionality . . . . .                         | 3          |
| 1.5 Problem Statement . . . . .                         | 4          |
| 1.6 Related Work . . . . .                              | 5          |
| 1.7 Research Objectives . . . . .                       | 6          |
| 1.8 Contributions . . . . .                             | 6          |
| <b>2 Adopted Methodology</b>                            | <b>7</b>   |
| 2.1 Introduction . . . . .                              | 7          |
| 2.2 Topology . . . . .                                  | 8          |
| 2.3 Mapping Scenarios . . . . .                         | 8          |
| 2.4 Remarks . . . . .                                   | 10         |
| <b>3 Controller Placement in SDN - Single Mapping</b>   | <b>11</b>  |
| 3.1 Mathematical Model . . . . .                        | 11         |
| 3.2 Mathematical Formalization . . . . .                | 11         |
| 3.2.1 Objective Function: . . . . .                     | 11         |
| 3.2.2 Constraints: . . . . .                            | 11         |
| <b>4 Controller Placement in SDN - Multiple Mapping</b> | <b>15</b>  |
| 4.1 Mathematical Model . . . . .                        | 15         |

|          |   |            |
|----------|---|------------|
| 4.2      | Mathematical Formalization . . . . .                    | 15         |
| 4.2.1    | Objective Function . . . . .                            | 15         |
| 4.2.2    | Constraints . . . . .                                   | 15         |
| <b>5</b> | <b>Results and Evaluation - Single Mapping</b>          | <b>19</b>  |
| 5.1      | Scenario Setup . . . . .                                | 19         |
| 5.2      | Analysis of Results . . . . .                           | 21         |
| 5.3      | Conclusions . . . . .                                   | 24         |
| <b>6</b> | <b>Results and Evaluation - Multiple Mapping</b>        | <b>25</b>  |
| 6.1      | Scenario Setup . . . . .                                | 25         |
| 6.2      | Analysis of Results . . . . .                           | 27         |
| 6.3      | Comparing Multiple Mapping and Single Mapping . . . . . | 30         |
| 6.4      | Conclusions . . . . .                                   | 32         |
| <b>7</b> | <b>Conclusions and Future Work</b>                      | <b>33</b>  |
|          | <b>References</b>                                       | <b>35</b>  |
| <b>A</b> | <b>Implementation</b>                                   | <b>A-1</b> |
| A.1      | Introduction . . . . .                                  | A-1        |
| A.2      | Building the Model . . . . .                            | A-1        |

---

# List of Figures

---

|     |  |    |
|-----|--|----|
| 1.1 | Functions in networking. . . . .   | 1  |
| 1.2 | SDN Layers: management, control and data planes. . . . .   | 3  |
| 1.3 | SDN Flow. . . . .  | 4  |
| 2.1 | SDN Topology: no failed link. . . . .  | 8  |
| 2.2 | SDN Topology with 1 failed link. . . . .   | 8  |
| 2.3 | SDN Topology with 2 failed links. . . . .  | 9  |
| 5.1 | Physical topologies used for analysis of results in single-mapping architectures. Large nodes are controllers, medium are switches, and small are non-used locations. Different color indicates switch-controller mapping. . . . .   | 20 |
| 5.2 | Results for Topology I, having high connectivity and relatively low number of possible locations for the controllers, using single-mapping. . . . .  | 22 |
| 5.3 | Results for Topology II, having high connectivity and relatively low number of possible locations for the controllers, using single-mapping. . . . .   | 23 |
| 5.4 | Results for Topology III, having high connectivity and relatively low number of possible locations for the controllers, using single-mapping. . . . .  | 24 |
| 6.1 | Physical topologies used for analysis of results in multiple-mapping architectures. Large nodes are controllers, medium are switches, and small are non-used locations. Different color indicates switch-controller mapping. . . . . | 26 |
| 6.2 | Results for Topology I, having high connectivity and relatively low number of possible locations for the controllers, using multiple-mapping. . . . .  | 28 |
| 6.3 | Results for Topology II, having high connectivity and relatively low number of possible locations for the controllers, using multiple-mapping. . . . .   | 29 |
| 6.4 | Results for Topology III, having high connectivity and relatively low number of possible locations for the controllers, using multiple-mapping. . . . .  | 30 |
| 6.5 | Scalability factor for Topology I, having high connectivity and relatively low number of possible locations for the controllers. . . . .   | 31 |
| 6.6 | Scalability factor for Topology II, having low connectivity and relatively low number of possible locations for the controllers. . . . .   | 31 |

|  |     |
|--|-----|
| 6.7 Scalability factor for Topology III, having low connectivity and relatively high number of possible locations for the controllers. . . . . | 31  |
| 7.1 Hypervisor-Based Overlay Networks. . . . .   | 34  |
| A.1 General structure of CPLEX solving model. . . . .  | A-1 |
| A.2 Main flow chart of a test (F is the number of scenarios). . . . .  | A-2 |

---

# List of Tables

---

|     |   |    |
|-----|---|----|
| 1.1 | SDN enabled switches. . . . .                             | 4  |
| 2.1 | Fault tolerance approaches [1]. . . . .                   | 7  |
| 2.2 | Failure types and comparison between controllers. . . . . | 7  |
| 3.1 | Known information for Single Mapping. . . . .             | 12 |
| 3.2 | Required variables for Single Mapping. . . . .            | 12 |
| 4.1 | Known information for Multiple Mapping. . . . .           | 16 |
| 4.2 | Required variables for Multiple Mapping. . . . .          | 16 |
| 5.1 | Input parameters - Single Mapping. . . . .                | 21 |
| 5.2 | Physical topology details - Single Mapping. . . . .       | 21 |
| 6.1 | Input parameters - Multiple Mapping. . . . .              | 27 |
| 6.2 | Physical topology details - Multiple Mapping. . . . .     | 27 |

---

# Nomenclature

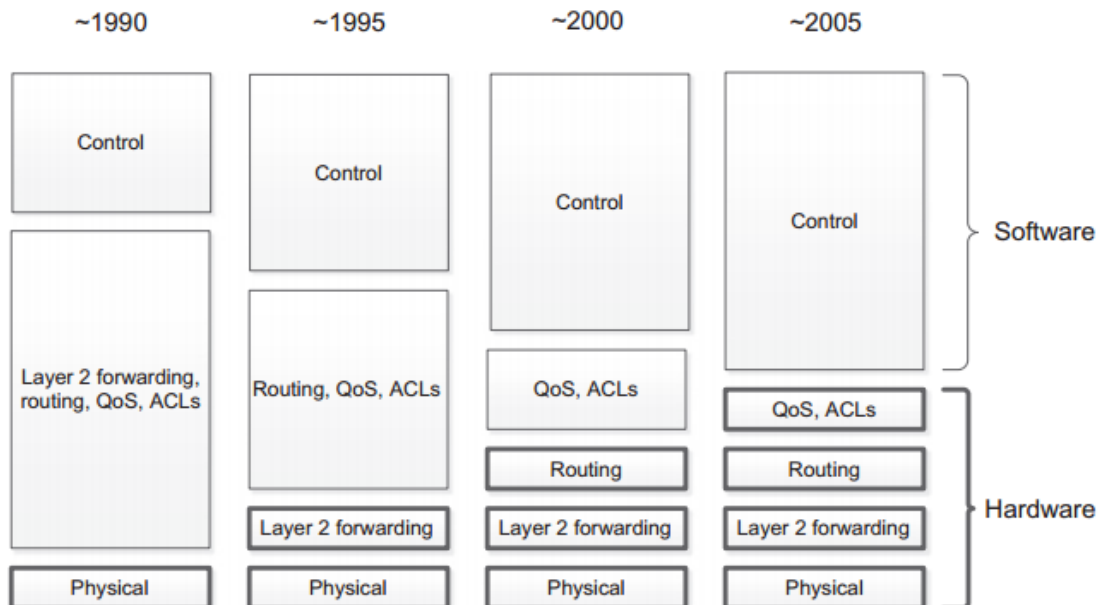
---

|       |                                 |
|-------|---------------------------------|
| SDN   | Software Defined Networkig      |
| NFV   | Network Function Virtualization |
| VTEP  | VXLAN Tunnel Endpoint           |
| VXLAN | Virtual Extensible LAN          |

Introduction

In the beginning of networking it was fairly common to have autonomous devices with all networking functionality built inside it, so the design of small networks was more straightforward. In Figure 1.1 it is easily shown that at the beginning the control and data planes were mostly implemented in software. Then, with the advance of new technologies, hardware became responsible for all data plane tasks while the control plane functionality continued to be done by software. All tasks relating data packet forwarding, in routers or switches, are placed under the data plane, while all tasks related to the overall topology and reachability of the network are placed under the control plane. More recently, Software Defined Networking (SDN) gained a lot of attention.[2]

Figure 1.1: Functions in networking.



### 1.1 SDN

Software Defined Networking is an emergent paradigm that offers a software-oriented network design, simplifying network management by decoupling the control logic from forwarding devices [3]. The control logic is the brain of the network that decides for best paths, prevents the outage and enforces the business logic into action. This is done in a centralized manner with the help of a computational device (server) instead of using the old distributed design, where every node in the network has the role of finding for best paths or recovering from failures.

### 1.2 SDN Layers

The SDN is composed of three planes: management, control, and data, which are shown in Figure 1.2. The management plane is responsible for defining the network policies, and it is connected to the control plane via northbound interfaces. The brain of SDN is the control plane, which interacts with the data plane using southbound interfaces like OpenFlow [4]. Data plane is mostly responsible of forwarding data packets in switching or routing manner.

In SDN, there are 3 different interfaces:

1. Northbound, a software interface that is responsible for communication between the management application and the controller module.
2. Southbound, a software interface that is responsible for communication between the controller and the networking device.
3. West-East, which is responsible for communication between controllers.

### 1.3 SDN Architecture

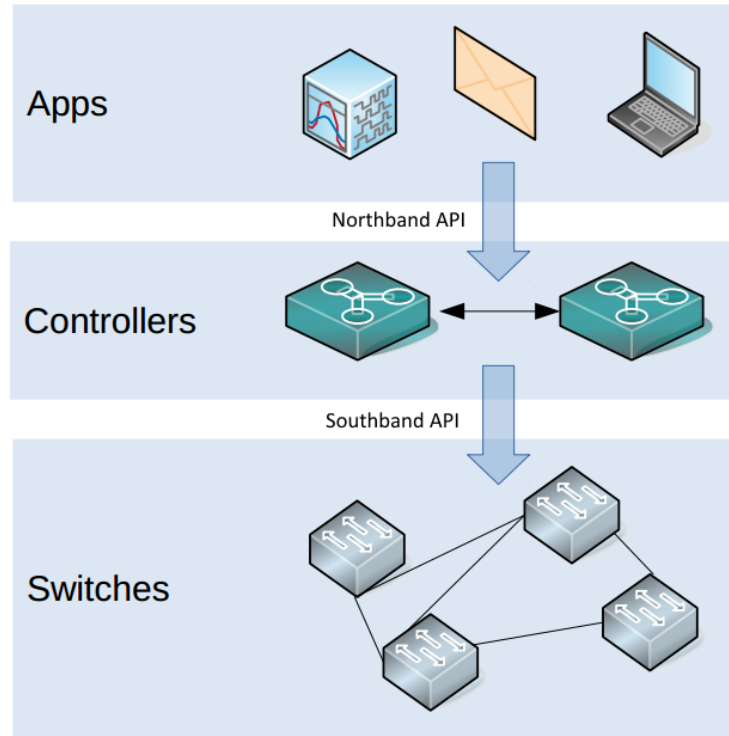
Most of the currently available controllers, like NOX [5] and Beacon [6], are physically centralized. Although a single controller offers a complete network-wide view, it represents a single point of failure and lacks both reliability and scalability [7]. For this reason multi-controller SDNs were developed [4], allowing the control plane to be physically distributed, but maintaining it logically centralized by synchronizing the network state among controllers [1]. Controllers of this type include OpenDaylight [8] and Kandoo [9]. Multi-controller SDNs are able to solve the main problems found in the centralized SDN, but new challenges are introduced, like network state synchronization and switch-controller mappings. Another main problem in multi-controller SDNs is the Controller Placement Problem (CPP) [10]. The problem has

## 1.4 SDN Functionality

---

proved to be NP-hard [11], and is one of the hottest topics in multi-controller SDNs [12, 11, 13, 14].

**Figure 1.2:** SDN Layers: management, control and data planes.



## 1.4 SDN Functionality

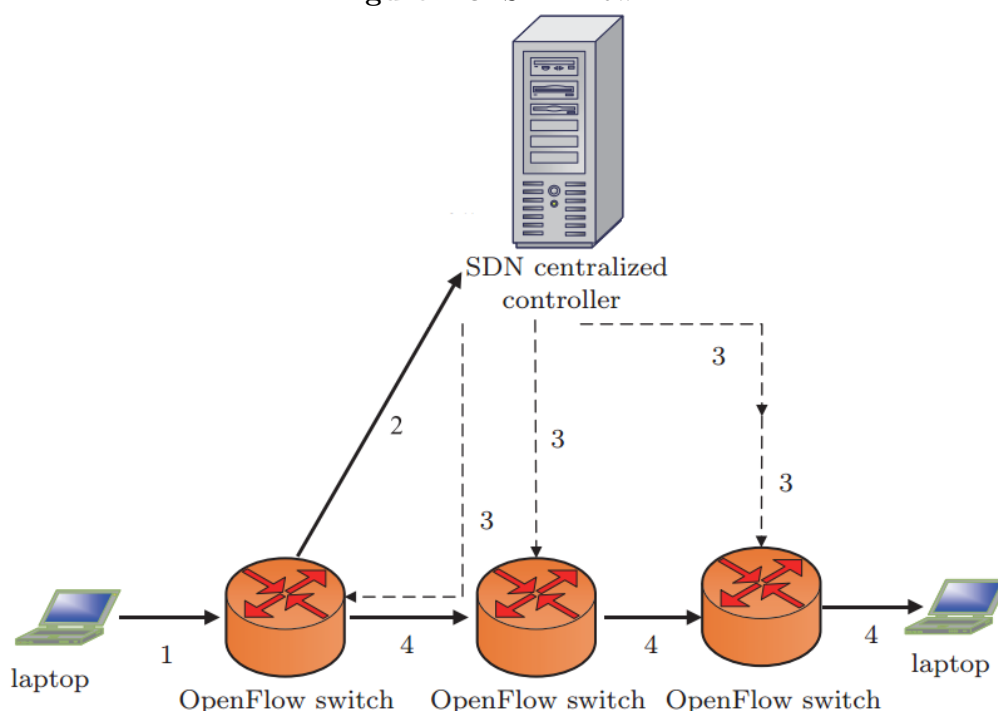
When a switch receives a new packet, it consults its forwarding rules in the flow table in order to determine how to handle the packet. If there is no match in the flow table, the packet is buffered temporarily and the switch initiates a **packet-in** message to the master controller. Reactively, the controller calculates the path for this packet and installs a new rule in the affected switches. Two major factors that influence the effectiveness of this process are: *i*) load at the controller; *ii*) propagation delay between the switch and the controller. These two constitute a single efficiency measure: the flow setup time, which will be twice the switch-controller propagation delay plus the queuing and processing time at controllers. The steps performed by a switch and a controller are shown in Figure 1.3. These are the following:

1. The first packet misses the flow-table entry.
2. The flow is forwarded to the controller so that a new flow entry, including an action on the particular flow, can be found.
3. Controller installs the new flow entry.

## 1.5 Problem Statement

---

Figure 1.3: SDN Flow.



4. A new designated flow path.

Nowadays there are SDN enabled products used for different networking landscapes. Table 1.1 illustrates some of the SDN enabled devices that can be used in different production environments [15].

**Table 1.1:** SDN enabled switches.

| Product      | Type   | Description               | Vendor  |
|--------------|--------|---------------------------|---------|
| Arista 7150  | Switch | Data center class chassis | Arista  |
| CX600 series | Router | Carrier Class Man Router  | Huawei  |
| MLX Series   | Router | Service Provider class    | Brocade |
| Rackswitch   | Switch | Data center Class Switch  | IBM     |

## 1.5 Problem Statement

As it was mentioned before, one of the main problems in multicontroller SDNs is the Controller Placement Problem (CPP). There are three main questions regarding the controller placement problem [16]:

1. How many controllers are needed?
2. Where should controllers be placed?

## 1.6 Related Work

---

### 3. Which metrics should be considered?

The main concern in SDN networks is deciding for the number of the controllers, while not letting too many controllers affect the deployment cost. Note, however, that reducing them too much will turn the control plane into a bottleneck.

The second concern in SDN is, therefore, the location of controllers. Network operators may plan the controller location from scratch, or may plan for the expansion of the network, usually having cost minimization in mind while meeting the growing number of users and services. In the expansion case this means avoiding disturbing the operation of the network. The focus of this work plan is the placement of controllers while considering resilience.

Another concern is the scalability. This is a major concern in any system and it is intensified in SDNs due to the introduction of new control plane traffic (configuration of switches by controllers). This adds extra concern for controllers which have to process many requests, forcing researchers to work more on scalability so that networks become more productive.

There are 3 different approaches to tackle the scaling problem in SDN networks. These act, respectively, on: *i*) data plane, *ii*) control plane, *iii*) hybrid [17]. The first approach is to reduce the overloading of controllers by transferring some jobs to forwarding devices. The second approach is to increase the computational power of controllers or use distributed controllers. The last one is the combination of the two previous approaches.

The CPP aims at deciding for the number of required controllers and where to place them [12], partitioning the network into subregions (domains), while considering some quality criteria and cost constraints [13, 18]. The CPP model discussed in this thesis incorporates the previously mentioned flow setup time, while presenting reliable and scalable solutions.

## 1.6 Related Work

The placement problem was mentioned for the first time in [12]. In fact, this problem is similar to the popular facility location problem, and is solved in the aforementioned article as K-center problem, to minimize the inter-plane propagation delay. In [11] the problem is extended to incorporate the capacity of the controllers. A new metric called expected percentage of control path loss is proposed in [19] to guarantee a reliable model. Cost, controller type, bandwidth, and other factors are considered in [13], and the expansion problem is considered in [14]. The problem is usually modeled in these articles as an integer programming problem.

Heuristic methods that incorporate switch migration can be found in [20]. In

## 1.7 Research Objectives

---

[21] a game model is also proposed. A comprehensive review of heuristic methods can be found in [22]. QoS-aware CPP is presented in [23] and solved using greedy and network partitioning algorithms. Recently, scalability and reliability issues in large-scale networks are considered in [18]. Clustering and genetic approaches are proposed, but these approaches are prone to sub-optimality.

## 1.7 Research Objectives

The main goal of this thesis is to define a mathematical model that minimizes metrics related to the deployment of controllers in SDN, while considering the scalability of the network. The work is done in the following steps:

1. Literature review of existing approaches for SDN controller placement problem.
2. Proposal of approaches for controller placement, while considering resilience and scalability, for single-mapping and multiple-mapping cases.
3. Implementation and evaluation proposals.

## 1.8 Contributions

While in [24] the authors make failure planning in a way to choose backup controllers and minimize two latencies, the latency from switch to the primary controller and to the backup controller, the approach followed in this thesis uses link failure scenarios and then decides for a controller placement that minimizes latency taking all failure scenarios into consideration. Besides such difference regarding protection, the scalability is also taken into account. As far as known those two concerns have not been addressed simultaneously in the literature. The single mapping (SM) case is addressed in Chapter 3, while the multiple mapping (MM) case addressed in Chapter 4. Chapters 5 and 6 evaluate the results.

As a result of the first step (SM) a paper has been presented in a conference:

- Mohammad Ashrafi, Noélia Correia and Faroq AL-Tam. A Scalable and Reliable Model for the Placement of Controllers in SDN Networks, Broadnet 2018.

---

## Adopted Methodology

---

### 2.1 Introduction

In the beginning of SDN, every switch was assigned to a single controller (single mapping) but this couldn't satisfy criteria like load balancing, failover and offloading of controllers. For example, to tackle failures different approaches are suggested in Table 2.1, with a brief comparison. Therefore, multiple mapping (each switch mapped to several controllers) is more accepted for tackling failures. In this thesis two kind of modeling for controller placement are tested: single mapping and multiple mapping. These integer or mixed integer linear programming models are introduced to CPLEX<sup>1</sup> optimizer that solves them.

**Table 2.1:** Fault tolerance approaches [1].

| Mechanism of fault tolerance | Fault type         | Solution  |
|------------------------------|--------------------|---|
| Controller replication       | Controller fault   | Ensure the switches-controllers communication without interruption    |
| Failure recovery             | Node or link fault | Switches recovery without controllers, recovery time within 50 ms     |
| Protection switching         | Node or link fault | Monitoring function, fault recovery time within 50 ms                 |
| Segment protection           | Node or link fault | Avoid controllers' participation in recovery process                  |
| Fast failover                | Node or link fault | Maximize network resiliency   |
| Packets modification         | Node or link fault | Carry backup route or fault message to change the forwarding policies |

To have a better understanding of how failure is tackled by different controllers, Table 2.2 shows a comparison between different controllers and their capabilities to tackle different kinds of failures.

**Table 2.2:** Failure types and comparison between controllers.

| Controller platform | Failure type             | Solution                                |
|---------------------|--------------------------|---|
| ONOS                | ONOS instances failure   | Redundant instances                     |
| ONIX                | Link or switch failure   | Backup paths Active replication         |
| PANE                | RoLink or switch failure | Forwarding policy reconfiguration       |
| SmartLight          | Controller failure       | Replicated shared database for recovery |

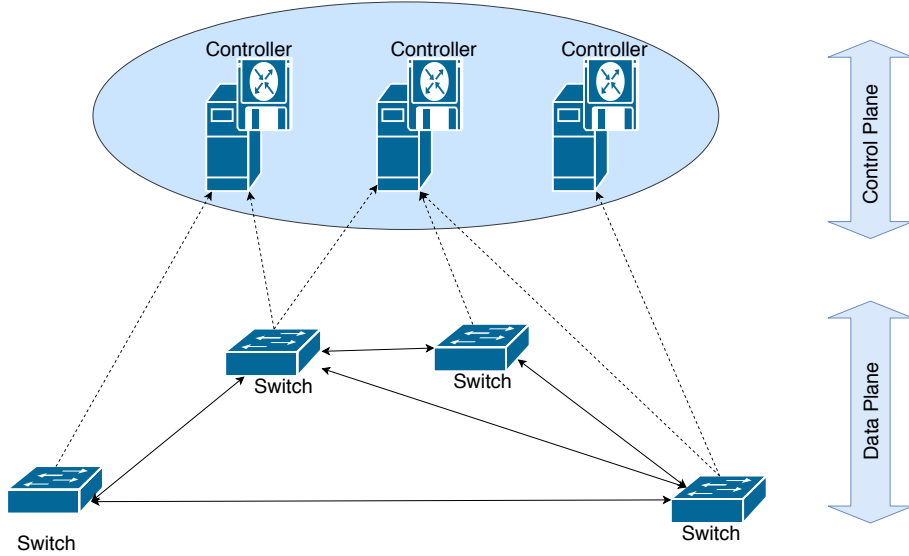
---

<sup>1</sup>IBM ILOG CPLEX Optimizer version 12.8.

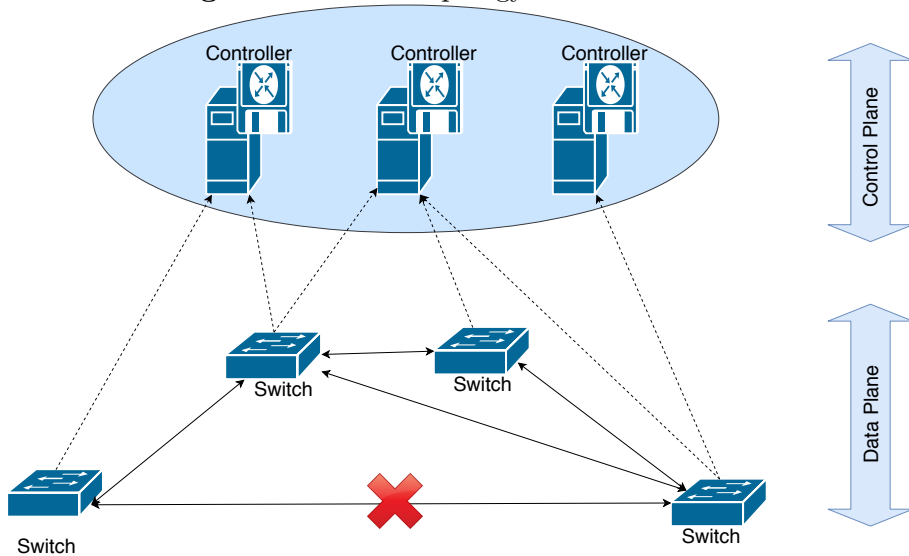
## 2.2 Topology

Keeping in mind that we are trying to introduce new metrics like scalability and failover, we have done this by creating different failure scenarios. In each scenario we introduce one or more failing links by removing it from original topology. Please note that switches refer here to layer 3 devices, as is commonly used in the SDN literature.

**Figure 2.1:** SDN Topology: no failed link.



**Figure 2.2:** SDN Topology with 1 failed link.

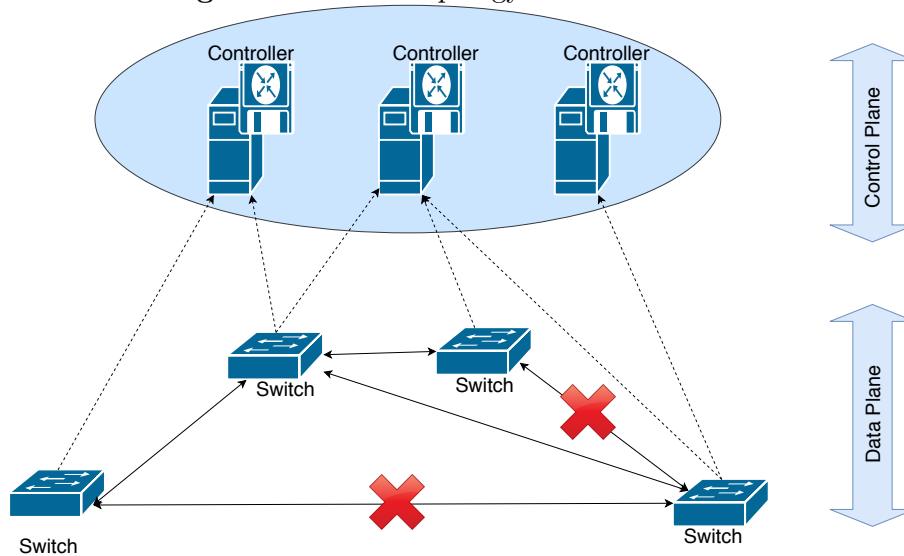


## 2.3 Mapping Scenarios

The following switch to controller mapping scenarios are expected to be considered:

## 2.3 Mapping Scenarios

**Figure 2.3:** SDN Topology with 2 failed links.



1. Single mapping (SM)
2. Multiple mapping (MM)
3. Multiple mapping with overlay topology - future work

For single mapping, every switch is connected to a single controller and there is no load sharing among controllers. The model provided in Chapter 3, whose analysis of results in Chapter 5, fits to single mapping. In Chapter 4, a proposed model for multiple mapping is discussed, which is capable of load sharing between controllers. That is, a single switch is controlled by multiple controllers making this a resilient approach.

Distributed controller is a key concept to tackle the single point of failure issue, making the network more resilient, and in case of failure traffic will flow around the failed node. Controllers reside on physical devices, which are subject to failures, while the software is also subject to failure due to poor design, bugs or overload because of scale [25]. These scenarios enforce the SDN designer to think of a logically centralized controller made of one or multiple controllers. The former, as it was mentioned, is a single point of failure which is not accepted in production environments. The latter will require complex approaches to maintain the global view of the network consistent.

As everybody knows “Many hands make light work.”, and this is exactly how a distributed controller works. It makes handling the load of requests easier compared to single controller, but it will introduce new challenges regarding the location of controllers, controller-switch mapping and delay. The latter is very vital because with high delay, between switch and controller, the overall performance of the net-

## 2.4 Remarks

---

work will decrease and the controller can not respond to requests in a timely manner [26].

## 2.4 Remarks

In the next chapter, optimal solution models will be found that place controllers on a SDN network or expand an existing one. Besides minimizing the cost of the network, various constrains like flow-setup latency and controller to controller connectivity will also be considered.

---

# Controller Placement in SDN - Single Mapping

---

## 3.1 Mathematical Model

In the following discussion the physical topology graph is assumed to be defined by  $\mathcal{G}(\mathcal{N}, \mathcal{L})$ , where  $\mathcal{N}$  is a set of physical nodes/locations and  $\mathcal{L}$  is a set of physical links. The remaining notation for known information and variables, used through this report, is presented in Tables 3.1 and 3.2, respectively.

## 3.2 Mathematical Formalization

### 3.2.1 Objective Function:

To ensure the linear scale up of the SDN network, the goal will be:

$$\text{Minimize } \delta + K_1 \frac{\Theta^{\text{TOTAL}}}{\Delta} + K_2 \frac{\Pi^{\text{TOTAL}}}{\Delta} \quad (3.1)$$

where  $\Delta$  is a big value. The primary goal is to minimize  $\delta$ , the scalability factor, and then to reduce latency. The factors  $K_1$  and  $K_2$  should be adapted according to switch-controller and inter-controller latency relevance. The motivation behind giving  $\delta$  more importance is that the provided solution for controller placement will be used for a relatively long period of time, during which traffic conditions may change. Therefore, the scalability of the solution is considered to be critical.

### 3.2.2 Constraints:

The following additional constraints must be fulfilled.

- Placement of controllers:

$$\sum_{\{n \in \mathcal{N}_c\}} \sigma_n^c = 1, \forall c \in \mathcal{C} \quad (3.2)$$

### 3.2 Mathematical Formalization

**Table 3.1:** Known information for Single Mapping.

| Term            | Description  |
|-----------------|--|
| $\mathcal{C}$   | Set of controllers.  |
| $\mathcal{N}_c$ | Possible places for controller $c \in \mathcal{C}$ , $\mathcal{N}_c \subseteq \mathcal{N}$ . |
| $h_c$           | Number of requests per second that can be handled by controller $c \in \mathcal{C}$ .        |
| $\mathcal{S}$   | Set of switches.   |
| $p_s$           | Number of requests not matching the lookup table of $s \in \mathcal{S}$ .                    |
| $\mathcal{F}$   | Set of physical link failure scenarios. Includes a scenario where all links are up.          |
| $\mathcal{L}_f$ | Set of physical links failing when scenario $f \in \mathcal{F}$ occurs.                      |

**Table 3.2:** Required variables for Single Mapping.

| Variable                          | Description  |
|-----------------------------------|--|
| $\sigma_n^c$                      | One if controller $c \in \mathcal{C}$ is placed at location $n \in \mathcal{N}_c$ .  |
| $\mu_{n_i, n_j, f, l}^{c_i, c_j}$ | One if link $l \in \mathcal{L} \setminus \mathcal{L}_f$ is used for inter-controller $c_i - c_j$ communication, located at nodes $n_i$ and $n_j$ respectively, when failure $f \in \mathcal{F}$ occurs.                            |
| $\beta_f^l$                       | One if link $l \in \mathcal{L} \setminus \mathcal{L}_f$ is used for inter-controller communication when failure $f \in \mathcal{F}$ occurs.  |
| $\gamma_f^{s, c}$                 | One if switch $s \in \mathcal{S}$ is assigned to controller $c \in \mathcal{C}$ when failure $f \in \mathcal{F}$ occur.  |
| $\lambda_{n, f}^{s, c}$           | One if switch $s \in \mathcal{S}$ is assigned to controller $c \in \mathcal{C}$ when failure $f \in \mathcal{F}$ occurs, and the controller is placed at location $n \in \mathcal{N}_c$ .  |
| $\phi_{f, l}^{s, c, n_i}$         | One if switch $s \in \mathcal{S}$ is assigned to controller $c \in \mathcal{C}$ located at nodes $n_i \in \mathcal{N}_c$ when failure $f \in \mathcal{F}$ , and uses link $l \in \mathcal{L} \setminus \mathcal{L}_f$ in its path. |
| $\delta$                          | Scalability factor, $0 \leq \delta \leq 1$ .   |
| $\Theta^{\text{TOTAL}}$           | Total latency, under any failure scenario.   |
| $\Pi^{\text{TOTAL}}$              | Total number of links used, under any failure scenario.  |

$$\begin{aligned}
 & \sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{src}(l) = n\}} \mu_{n_i, n_j, f, l}^{c_i, c_j} - \sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{dst}(l) = n\}} \mu_{n_i, n_j, f, l}^{c_i, c_j} = \\
 & = \begin{cases} \sigma_{n_i}^{c_i}, & \text{if } n = n_i \\ -\sigma_{n_j}^{c_j}, & \text{if } n = n_j, \forall c_i, c_j \in \mathcal{C}, \forall f \in \mathcal{F}, \forall n_i \in \mathcal{N}_{c_i}, \forall n_j \in \mathcal{N}_{c_j}, \forall n \in \mathcal{N} \\ 0, & \text{otherwise} \end{cases} \quad (3.3)
 \end{aligned}$$

Constraints (3.2) ensure a single location for each controller  $c \in \mathcal{C}$ , while constraints (3.3) ensure that there will be a path between every pair of controllers, under any failure scenario, while considering their location. These paths are used for state synchronization among controllers.

In [27] it is stated that the controller load can be reduced, achieving load balance

### 3.2 Mathematical Formalization

---

among neighboring controllers, if controllers communicate only with its local neighbors. Therefore, the paths from any controller, towards all the other controllers, should share as many links as possible (leads to a bus logical topology). This is ensured by the following constraints.

$$\beta_f^l \geq \mu_{n_i, n_j, f, l}^{c_i, c_j}, \forall c_i, c_j \in \mathcal{C}, \forall n_i, n_j \in \mathcal{N}_c, \forall f \in \mathcal{F}, \forall l \in \mathcal{L} \quad (3.4)$$

$$\Pi^{\text{TOTAL}} = \sum_{\forall f \in \mathcal{F}} \sum_{\forall l \in \mathcal{L}} \beta_f^l \times 1/2 \quad (3.5)$$

where  $\Pi^{\text{TOTAL}}$ , counting for the highest number of end-to-end hops in inter-controller communication, is to be included in the objective function.

– Switch to controller mapping:

$$\sum_{\{c \in \mathcal{C}\}} \gamma_f^{s,c} = 1, \forall s \in \mathcal{S}, \forall f \in \mathcal{F} \quad (3.6)$$

$$\sum_{\{s \in \mathcal{S}\}} \gamma_f^{s,c} \times p_s \leq h_c \times \delta, \forall c \in \mathcal{C}, \forall f \in \mathcal{F} \quad (3.7)$$

Constraints (3.6) ensure single mapping and constraints (3.7) avoid the overload of controllers, while ensuring scalability regarding future switch migrations (triggered to deal with load fluctuations) due to the use of  $\delta$ , which is to be included in the objective function too. Again, the multiple failure scenarios are taken into consideration.

– Switch-controller latency:

$$\lambda_{n,f}^{s,c} \geq \gamma_f^{s,c} + \sigma_n^c - 1, \forall c \in \mathcal{C}, \forall n \in \mathcal{N}_c, \forall f \in \mathcal{F}, \forall s \in \mathcal{S} \quad (3.8)$$

$$\begin{aligned} & \sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{src}(l)=n\}} \phi_{f,l}^{s,c,ni} - \sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{dst}(l)=n\}} \phi_{f,l}^{s,c,ni} = \\ & = \begin{cases} \lambda_{n,f}^{s,c}, & \text{if } n = \text{loc}(s) \\ -\lambda_{n,f}^{s,c}, & \text{if } n = n_i \\ 0, & \text{otherwise} \end{cases}, \forall s \in \mathcal{S}, \forall c \in \mathcal{C}, \forall f \in \mathcal{F}, \forall n_i \in \mathcal{N}_c, \forall n \in \mathcal{N} \end{aligned} \quad (3.9)$$

Constraints (3.9) ensure switches are placed near to the corresponding controller, and  $\text{loc}(s)$  is the location of switch  $s$ . The total latency is obtained by:

$$\Theta^{\text{TOTAL}} = \sum_{\{s \in \mathcal{S}\}} \sum_{\{c \in \mathcal{C}\}} \sum_{\{f \in \mathcal{F}\}} \sum_{\{l \in \mathcal{L}\}} \sum_{\{n \in \mathcal{N}_c\}} \phi_{f,l}^{s,c,n} \quad (3.10)$$

### 3.2 Mathematical Formalization

---

The  $\Theta^{\text{TOTAL}}$  is included in the objective function for latency minimization.

– Non-negativity assignment to variables:

$$0 \leq \delta \leq 1; \sigma_n^c, \mu_{n_i, n_j, f, l}^{c_i, c_j}, \beta_f^l, \gamma_f^{s, c}, \lambda_{n, f}^{s, c}, \phi_{f, l}^{s, c, ni} \in \{0, 1\}; \Theta^{\text{TOTAL}}, \Pi^{\text{TOTAL}} \in \mathfrak{R}^+. \quad (3.11)$$

This model assumes that the physical layer is not disconnectable under a single physical link failure. The CPLEX<sup>1</sup> optimizer has been used to solve the problem instances discussed later in Chapter 5.

---

<sup>1</sup>IBM ILOG CPLEX Optimizer version 12.8.

---

# Controller Placement in SDN - Multiple Mapping

---

## 4.1 Mathematical Model

Let us define  $\mathcal{G}(\mathcal{N}, \mathcal{L})$  as the physical topology graph, where  $\mathcal{N}$  is a set of physical nodes/locations and  $\mathcal{L}$  is a set of physical links. The remaining notation for known information and variables, used through this report, is presented in Tables 4.1 and 4.2 respectively.

## 4.2 Mathematical Formalization

### 4.2.1 Objective Function

To ensure the linear scale up of the SDN network the goal will be:

$$\text{Minimize } \delta + K_1 \frac{\Theta^{\text{MAX}}}{\Delta} + K_2 \times \frac{\Pi^{\text{MAX}}}{\Delta} \quad (4.1)$$

where  $\Delta$  is a big value. The primary goal is to minimize  $\delta$ , and then to reduce latency.  $K_1$  vs  $K_2$  should be adapted according to switch-controller and inter-controller latency relevance.

### 4.2.2 Constraints

The following additional constraints must be fulfilled.

- Placement of controllers:

$$\sum_{\{n \in \mathcal{N}_c\}} \sigma_n^c = 1, \forall c \in \mathcal{C} \quad (4.2)$$

## 4.2 Mathematical Formalization

| Term            | Description  |
|-----------------|--|
| $\mathcal{C}$   | Set of controllers.  |
| $\mathcal{N}_c$ | Possible places for controller $c \in \mathcal{C}$ , $\mathcal{N}_c \subseteq \mathcal{N}$ . |
| $h_c$           | Number of requests per second that can be handled by controller $c \in \mathcal{C}$ .        |
| $\mathcal{S}$   | Set of switches.   |
| $p_s$           | Number of requests not matching the lookup table of $s \in \mathcal{S}$ .                    |
| $\mathcal{F}$   | Set of physical link failure scenarios. Includes a scenario where all links are up.          |
| $\mathcal{L}_f$ | Set of physical links failing when scenario $f \in \mathcal{F}$ occurs.                      |

| Variable                          | Description  |
|-----------------------------------|--|
| $\sigma_n^c$                      | One if controller $c \in \mathcal{C}$ is placed at location $n \in \mathcal{N}_c$ .  |
| $\mu_{n_i, n_j, l, f}^{c_i, c_j}$ | One if link $l \in \mathcal{L} \setminus \mathcal{L}_f$ is used for inter-controller $c_i - c_j$ communication, when controllers are placed at $n_i$ and $n_j$ , respectively, and failure $f \in \mathcal{F}$ occurs. |
| $\beta_f^l$                       | One if link $l \in \mathcal{L} \setminus \mathcal{L}_f$ is used for inter-controller communication when failure $f \in \mathcal{F}$ occurs.  |
| $\gamma_f^{s, c}$                 | Fraction of flow from switch $s \in \mathcal{S}$ to controller $c \in \mathcal{C}$ when failure $f \in \mathcal{F}$ occurs.  |
| $\phi_{f, l}^{s, c}$              | Fraction of flow from switch $s \in \mathcal{S}$ to controller $c \in \mathcal{C}$ , flowing at link $l \in \mathcal{L} \setminus \mathcal{L}_f$ when failure $f \in \mathcal{F}$ occurs.                              |
| $\omega_{s, f}^{c, n}$            | Fraction of flow arriving to controller $c \in \mathcal{C}$ , if located at node $n \in \mathcal{N}$ , coming from switch $s \in \mathcal{S}$ when failure $f \in \mathcal{F}$ occurs.                                 |
| $\Theta^{\text{MAX}}$             | Highest switch-controller latency, under any failure scenario.   |
| $\Pi^{\text{MAX}}$                | Highest number of links used for inter controller communication, under any failure scenario.   |

$$\begin{aligned}
& \sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{src}(l) = n\}} \mu_{n_i, n_j, l, f}^{c_i, c_j} - \sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{dst}(l) = n\}} \mu_{n_i, n_j, l, f}^{c_i, c_j} = \\
& = \begin{cases} \sigma_{n_i}^{c_i}, & \text{if } n = n_i \\ -\sigma_{n_j}^{c_j}, & \text{if } n = n_j \\ 0, & \text{otherwise} \end{cases}, \forall c_i, c_j \in \mathcal{C}, \\
& , \forall n_i \in \mathcal{N}_{c_i}, \forall n_j \in \mathcal{N}_{c_j}, \forall f \in \mathcal{F}, \forall n \in \mathcal{N} \quad (4.3)
\end{aligned}$$

Constraints (4.2) ensure a single location for each controller  $c \in \mathcal{C}$ , while constraints (4.3) ensure that there will be a path between every pair of controllers, under any failure scenario and considering their location. For controllers to communicate just with their local neighbors, the paths from any controller, towards all the other controllers, should share as many links as possible. The following is required:

## 4.2 Mathematical Formalization

---

$$\beta_f^l \geq \mu_{n_i, n_j, l, f}^{c_i, c_j}, \forall c_i, c_j \in \mathcal{C}, \forall n_i \in \mathcal{N}_{c_i}, \forall n_j \in \mathcal{N}_{c_j}, \\ , \forall l \in \mathcal{L}, \forall f \in \mathcal{F} \quad (4.4)$$

$$\Pi^{\text{MAX}} \geq \frac{1}{2} \sum_{\forall l \in \mathcal{L}} \beta_f^l, \forall f \in \mathcal{F} \quad (4.5)$$

and  $\Pi^{\text{MAX}}$ , counting for the highest number of hops between any two controllers, should be included in the objective function.

– Switch to controller mapping:

$$\sum_{\{c \in \mathcal{C}\}} \gamma_f^{s,c} = 1, \forall s \in \mathcal{S}, \forall f \in \mathcal{F} \quad (4.6)$$

$$\sum_{\{s \in \mathcal{S}\}} \gamma_f^{s,c} \times p_s \leq h_c \times \delta, \forall c \in \mathcal{C}, \forall f \in \mathcal{F} \quad (4.7)$$

where constraints (4.6) ensure 100% delivery of switch requests to controllers, and constraints (4.7) avoid the overload of controllers, while ensuring scalability regarding future multi-mapping reassignments and/or switch migration (triggered to deal with load fluctuations) due to the inclusion of  $\delta$  used by the objective function. Again, the multiple failure scenarios are taken into consideration.

– Switch-controller latency:

$$\sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{src}(l) = n\}} \phi_{l,f}^{s,c,n_i} - \sum_{\{l \in \mathcal{L} \setminus \mathcal{L}_f : \text{dst}(l) = n\}} \phi_{l,f}^{s,c,n_i} = \\ = \begin{cases} \omega_{s,f}^{c,n} & \text{if } n = \text{loc}(s) \\ -\omega_{s,f}^{c,n} & \text{if } n = n_i \\ 0, & \text{otherwise} \end{cases}, \forall s \in \mathcal{S}, \forall c \in \mathcal{C}, \forall f \in \mathcal{F}, \\ , \forall n_i \in \mathcal{N}_c, \forall n \in \mathcal{N} \quad (4.8)$$

where  $\text{loc}(s)$  is the location of switch  $s$ . The  $\omega_{s,f}^{c,n}$  variables are limited by:

$$\sum_{\{s \in \mathcal{S}\}} \omega_{s,f}^{c,n} \leq \sigma_n^c \times \Delta, \forall c \in \mathcal{C}, \forall n \in \mathcal{N}_c, \forall f \in \mathcal{F} \quad (4.9)$$

## 4.2 Mathematical Formalization

---

$$\sum_{\{n \in \mathcal{N}_c\}} \omega_{s,f}^{c,n} = \gamma_f^{s,c}, \forall s \in \mathcal{S}, \forall c \in \mathcal{C}, \forall f \in \mathcal{F} \quad (4.10)$$

Switch-controller communication is ensured, under any failure scenario, according to placement of controllers. The highest latency is obtained by:

$$\Theta^{\text{MAX}} \geq \sum_{\{l \in \mathcal{L}\}} \sum_{\{n_i \in \mathcal{N}_c\}} \phi_{l,f}^{s,c,n_i}, \forall s \in \mathcal{S}, \forall c \in \mathcal{C}, \forall f \in \mathcal{F} \quad (4.11)$$

The  $\Theta^{\text{MAX}}$  is included at the objective function for latency minimization.

– Non-negativity assignment to variables:

$$\begin{aligned} 0 \leq \delta, \gamma_f^{s,c}, \phi_{f,l}^{s,c}, \omega_{s,f}^{c,n} \leq 1; \\ \sigma_n^c, \mu_{f,l}^{c_i,c_j}, \beta_f^l \in \{0, 1\}; \Theta^{\text{MAX}}, \Pi^{\text{MAX}} \in \mathfrak{R}^+. \end{aligned} \quad (4.12)$$

This model assumes that the physical layer is not disconnectable under any single physical link failure.

---

## Results and Evaluation - Single Mapping

---

### 5.1 Scenario Setup

The values for input parameters, used by the optimizer, are displayed in Table 5.1. Different failure cases were used to evaluate the model under the three topologies shown in Figure 5.1. A case relates to single link failure (no two links fail at the same time in each scenario), while the other relates to multiple link failure scenarios. Two percentages for affected links were tested. More specifically:

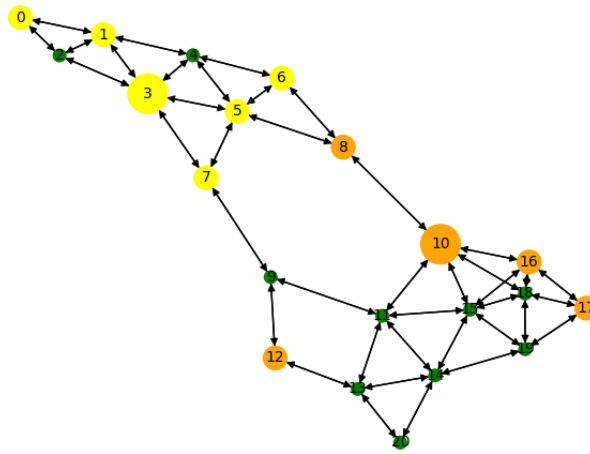
**Case I:** Single link failure scenarios, where  $\cup_{f \in \mathcal{F}} \mathcal{L}_f$  affects a total of 5% (a) or 15% (b) of all the links;

**Case II:** Two or more links failing simultaneously, in each failure scenario, where  $\cup_{f \in \mathcal{F}} \mathcal{L}_f$  affects a total of 5% (a) or 15% (b) of all the links.

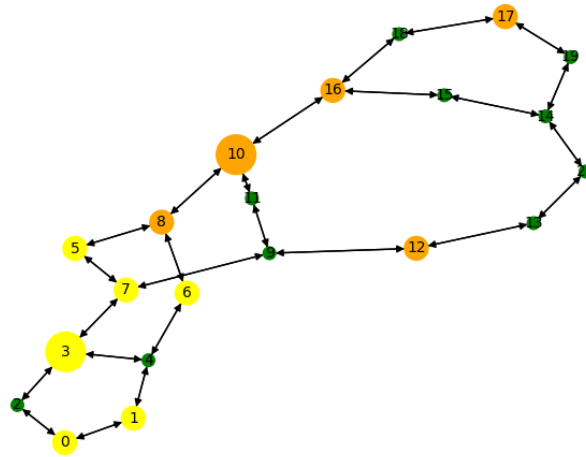
That is, in case Ia 5% of the links may fail, but no two links fail at the same time, while in case IIa there is a 0.5 probability for any two of the failed links to go down at the same time, which may lead to failure scenarios where two or more links fail simultaneously. In cases Ib and IIb, 15% of the links may fail.

## 5.1 Scenario Setup

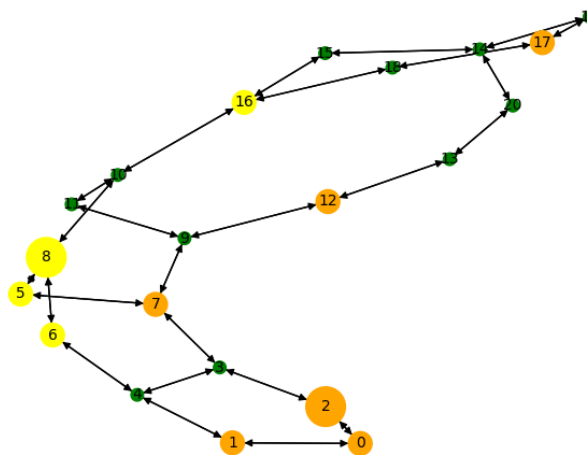
---



(a) Topology I. #Controllers=2 #Switches=9



(b) Topology II. #Controllers=2 #Switches=9



(c) Topology III. #Controllers=2 #Switches=9

**Figure 5.1:** Physical topologies used for analysis of results in single-mapping architectures. Large nodes are controllers, medium are switches, and small are non-used locations. Different color indicates switch-controller mapping.

## 5.2 Analysis of Results

---

**Table 5.1:** Input parameters - Single Mapping.

| Parameter | Value      |
|-----------|------------|
| $K_1$     | 0.5        |
| $K_2$     | 0.5        |
| $p_s$     | [40, 100]  |
| $h_s$     | [500, 600] |

**Table 5.2:** Physical topology details - Single Mapping.

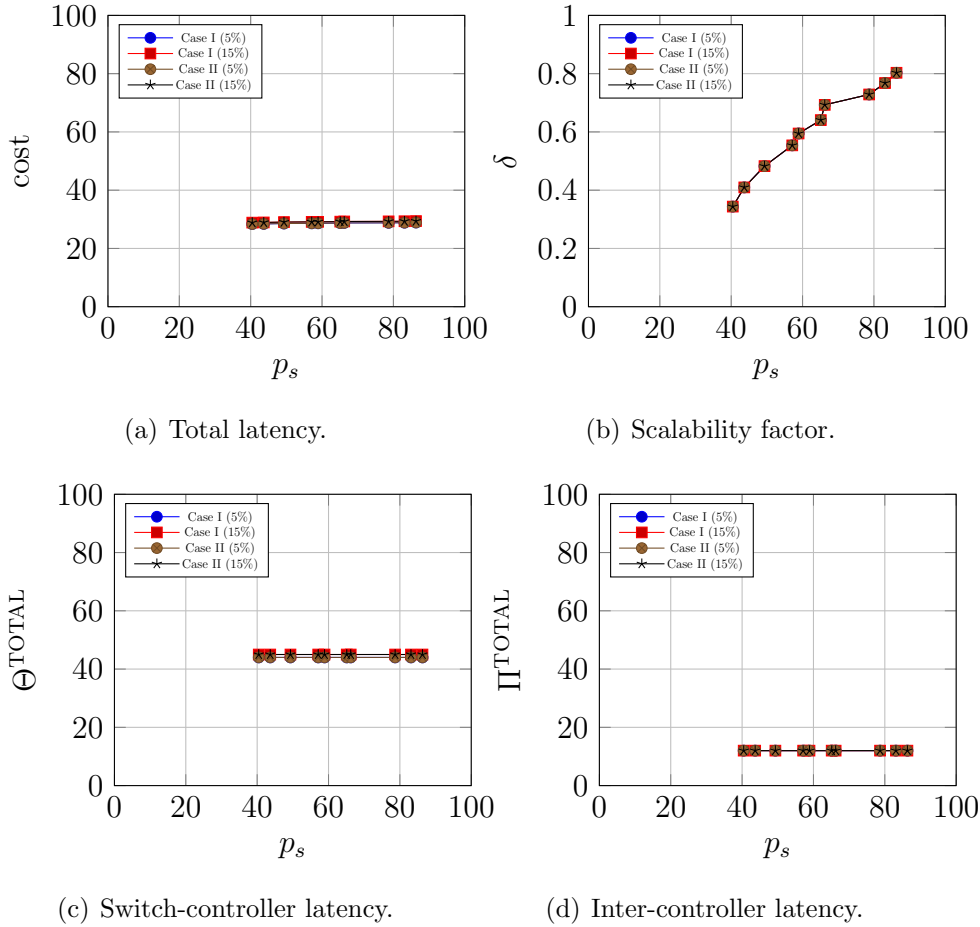
| Topology     | # Nodes | # Links | # Possible controller locations |
|--------------|---------|---------|---------------------------------|
| Topology I   | 21      | 40      | 4                               |
| Topology II  | 21      | 25      | 4                               |
| Topology III | 21      | 25      | 9                               |

## 5.2 Analysis of Results

### Topology I:

Results for this topology are shown in Figure 5.2. This is the most dense topology, with relatively small number of possible locations for controllers. The scalability factor  $\delta$  increases linearly as the number of **packet-in** messages increases, meaning that the gap between controller loads and capacity is reducing due to more packet-in messages arriving. The latency is relatively low, when compared with other topologies, because this topology has more links. For different failing scenarios the model presents similar results which is an indicator that the model is able to effectively deal with different failure scenarios.

## 5.2 Analysis of Results

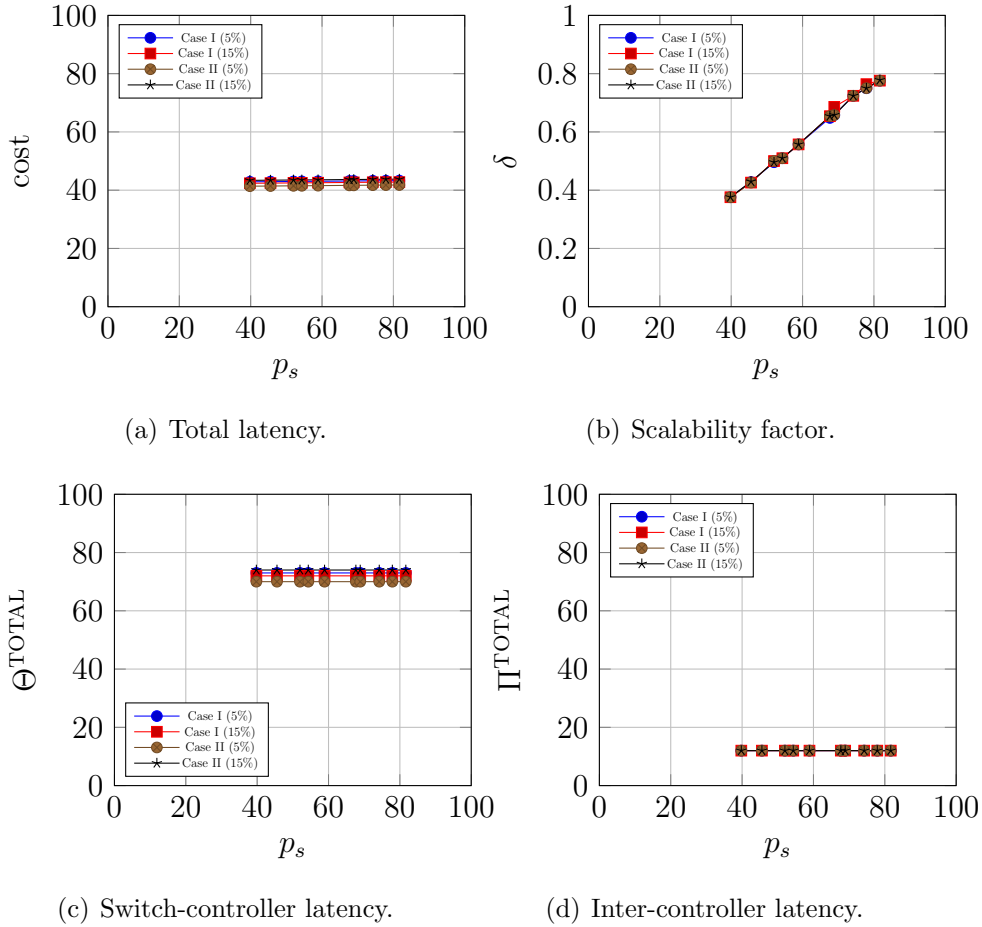


**Figure 5.2:** Results for Topology I, having high connectivity and relatively low number of possible locations for the controllers, using single-mapping.

### Topology II:

Results for this topology are depicted in Figure 5.3. This topology presents less links than Topology I, but the number of possible locations is kept similar. The main difference regarding these results is that the switch-controller latency has increased while the model keeps the inter-controller communication the same as in Topology I.

## 5.2 Analysis of Results

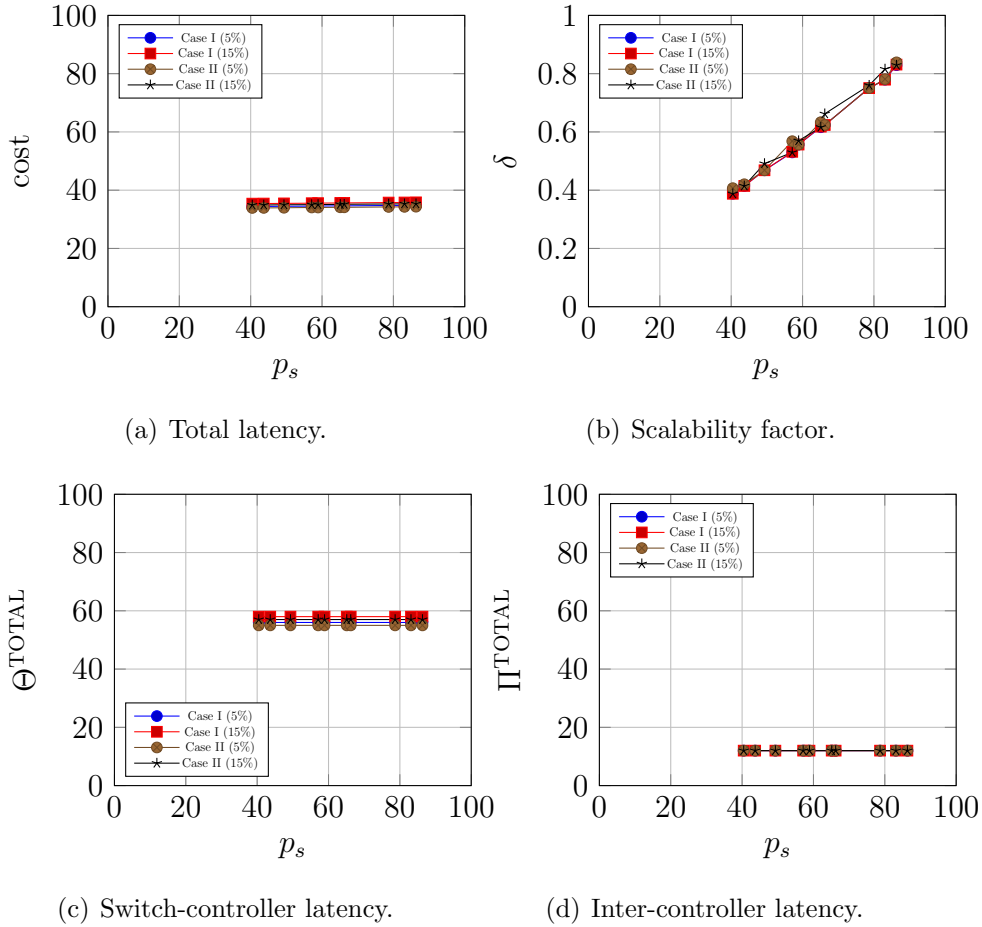


**Figure 5.3:** Results for Topology II, having high connectivity and relatively low number of possible locations for the controllers, using single-mapping.

### Topology III:

Results for this topology are shown in Figure 5.4. This topology also presents less links than Topology I, but the number of possible locations for each controller has increased. In this case it is possible to observe that the total latency significantly decreases due to better results for switch-controller latency. Therefore, the model was able to compensate the reduced number of links, finding adequate places for controllers that lead to lower latency. Results for all scenarios are the same which indicates that the model is able to deal with different failure scenarios without affecting the switch-controller latency, inter-controller latency, and scalability.

### 5.3 Conclusions



**Figure 5.4:** Results for Topology III, having high connectivity and relatively low number of possible locations for the controllers, using single-mapping.

### 5.3 Conclusions

Results show that scalability is ensured under different failure scenarios, while latency increase can be compensated through more freedom in controller's locations. The model also serves adequately multiple failure scenarios, presenting similar results for more critical failure scenarios and less critical ones.

In general, results show that the model is able to keep scalability ( $\delta$ ) while considering failure scenarios, ensuring load balancing among controllers. The latency may increase when less network connectivity decreases, but this might be avoided if more possible locations for controllers are allowed. Results are similar for Cases Ia, Ib, IIa and IIb, meaning that the model makes a controller placement that serves adequately multiple failure scenarios.

---

## Results and Evaluation - Multiple Mapping

---

### 6.1 Scenario Setup

Similarly to single mapping, randomly generated topologies have been used to evaluate the multiple mapping approach, and these are shown in Figure 6.1. The values for input parameters, used by the optimizer, are displayed in Table 6.1. Different failure cases were used to evaluate the model. A case relates to single link failure (no two links fail at the same time in each scenario), while the other relates to multiple link failure scenarios. Two percentages for affected links were tested. More specifically:

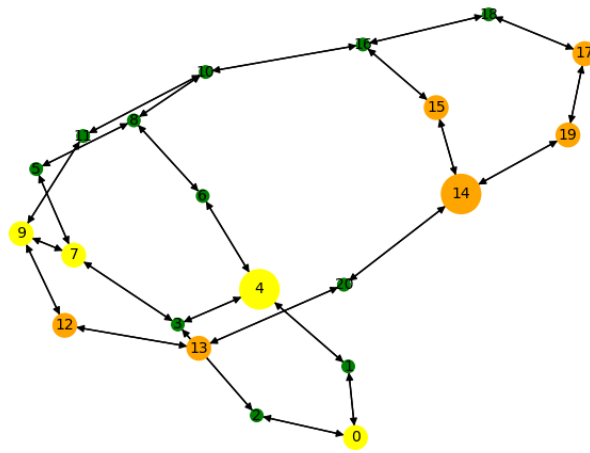
**Case I:** Single link failure scenarios, where  $\cup_{f \in \mathcal{F}} \mathcal{L}_f$  affects a total of 5% (a) or 15% (b) of all the links;

**Case II:** Two or more links failing simultaneously, in each failure scenario, where  $\cup_{f \in \mathcal{F}} \mathcal{L}_f$  affects a total of 5% (a) or 15% (b) of all the links.

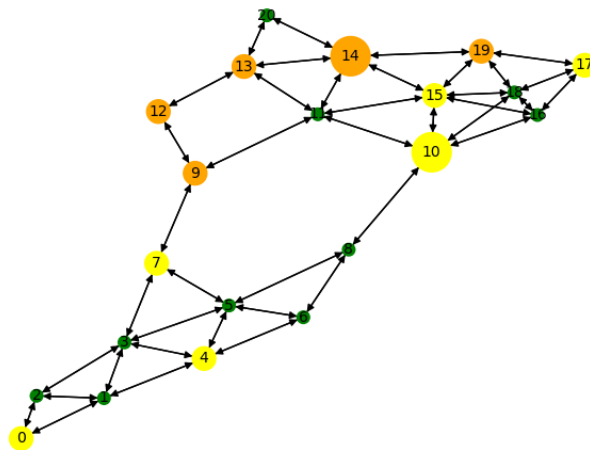
In case Ia, 5% of the links may fail, but no two links fail at the same time, while in case IIa there is a 0.5 probability for any two of the failed links to go down at the same time, which may lead to failure scenarios where two or more links fail simultaneously. In cases Ib and IIb, 15% of the links may fail.

## 6.1 Scenario Setup

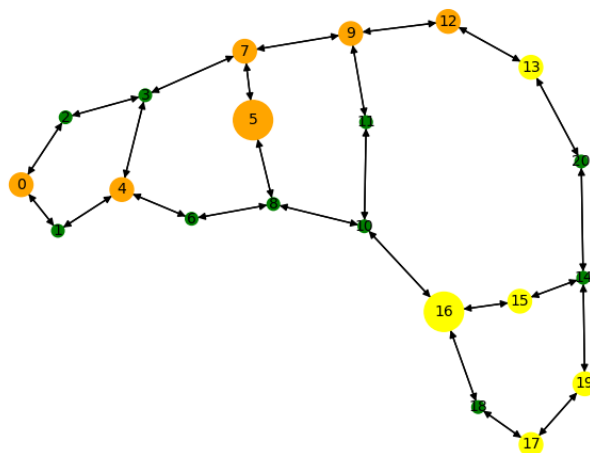
---



(a) Topology I. #Controllers=2 #Switches=9



(b) Topology II. #Controllers=2 #Switches=9



(c) Topology III. #Controllers=2 #Switches=9

**Figure 6.1:** Physical topologies used for analysis of results in multiple-mapping architectures. Large nodes are controllers, medium are switches, and small are non-used locations. Different color indicates switch-controller mapping.

## 6.2 Analysis of Results

---

**Table 6.1:** Input parameters - Multiple Mapping.

| Parameter | Value      |
|-----------|------------|
| $K_1$     | 0.5        |
| $K_2$     | 0.5        |
| $p_s$     | [40, 100]  |
| $h_s$     | [500, 600] |

**Table 6.2:** Physical topology details - Multiple Mapping.

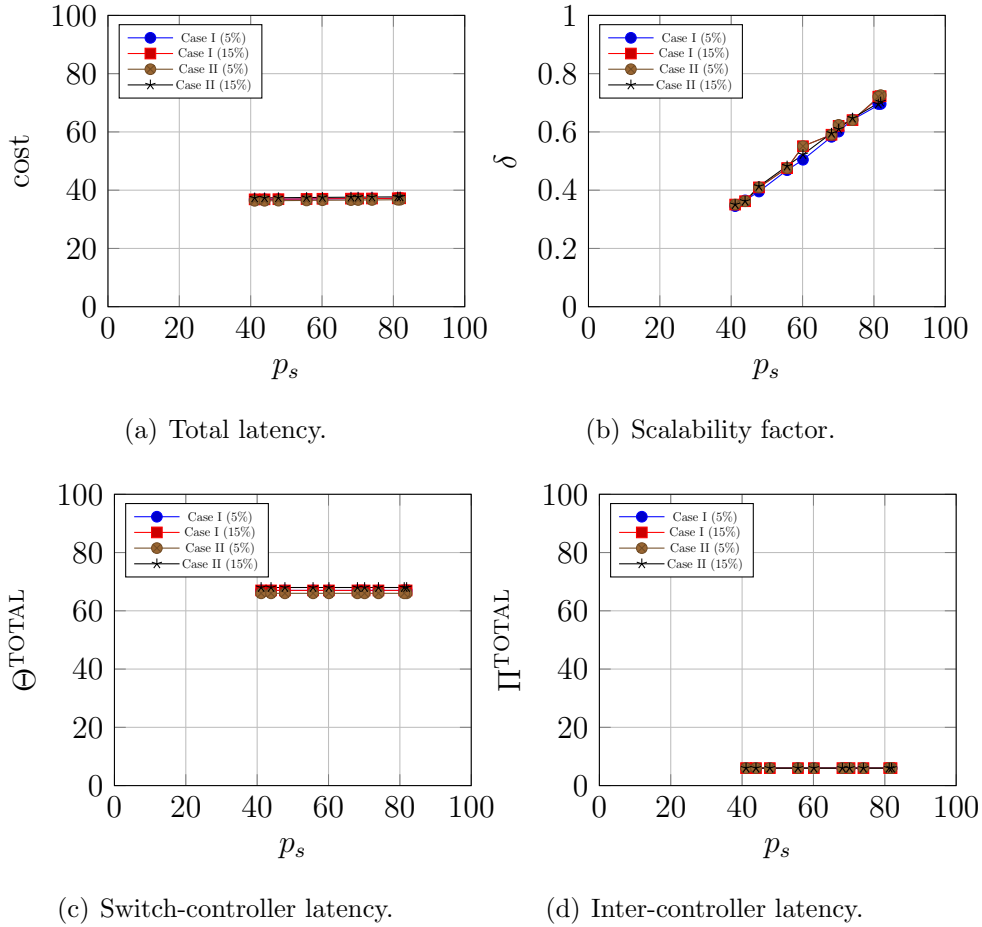
| Topology     | # Nodes | # Links | # Possible controller locations |
|--------------|---------|---------|---------------------------------|
| Topology I   | 21      | 40      | 4                               |
| Topology II  | 21      | 25      | 4                               |
| Topology III | 21      | 25      | 9                               |

## 6.2 Analysis of Results

### Topology I:

Results for this topology are shown in Figure 6.2. This is the most dense topology, with relatively small number of possible locations for controllers. The scalability factor  $\delta$  increases linearly as the number of `packet-in` messages increases, meaning that the gap between controller loads and capacity is reducing due to more packet-in messages arriving. The latency is relatively low, when compared with other topologies, because this topology has more links. For different failing scenarios the model presents similar results which is an indicator that the model is able to effectively deal with different failure scenarios. But inter-controller communication and switch-controller communication are very low since there are more connection between devices. For different failing scenarios the model has the same results which is an indicator that the model is effective under different failure scenarios, while keeping the same latency for inter-controller and switch-controller.

## 6.2 Analysis of Results

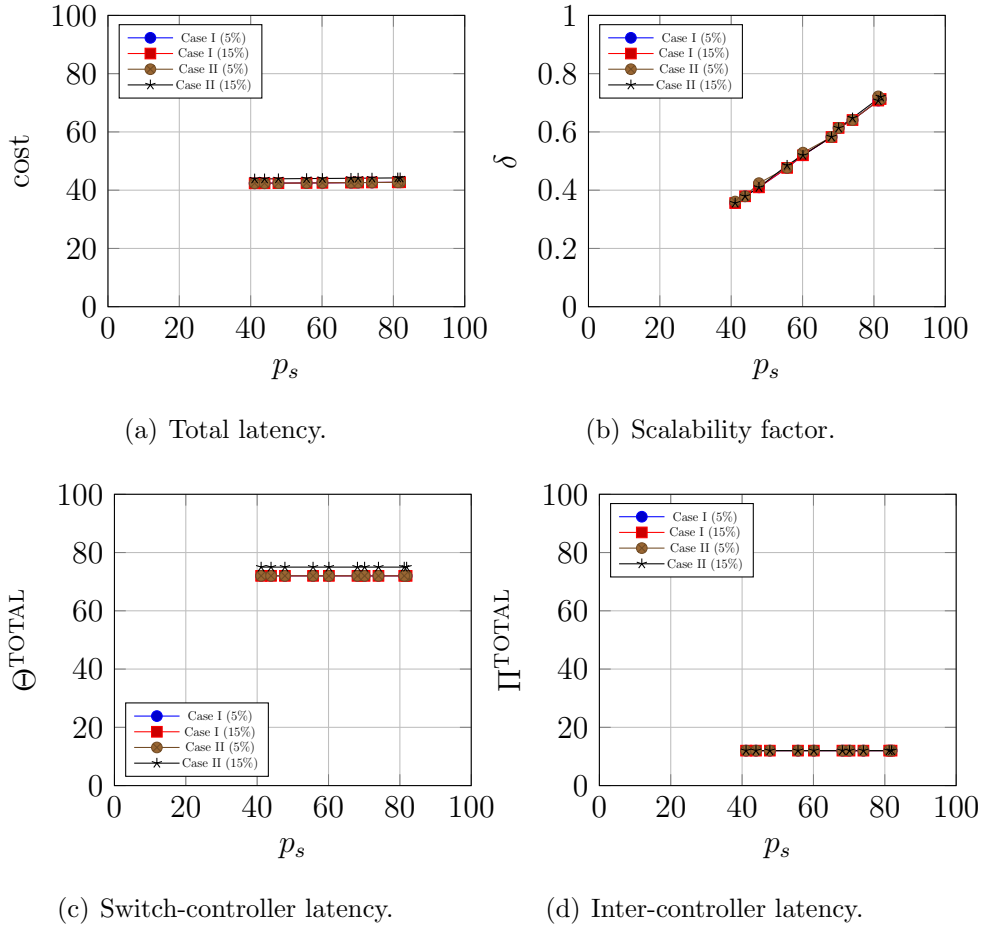


**Figure 6.2:** Results for Topology I, having high connectivity and relatively low number of possible locations for the controllers, using multiple-mapping.

### Topology II:

Results for this topology are depicted in Figure 6.3. This topology presents less links than Topology I, but the number of possible locations is kept similar. The main difference regarding these results is that the switch-controller latency has increased while the model keeps the inter-controller communication the same as in Topology I. The switch-controller latency has increased because this topology is less dense, although this has not affected inter-controller latency.

## 6.2 Analysis of Results

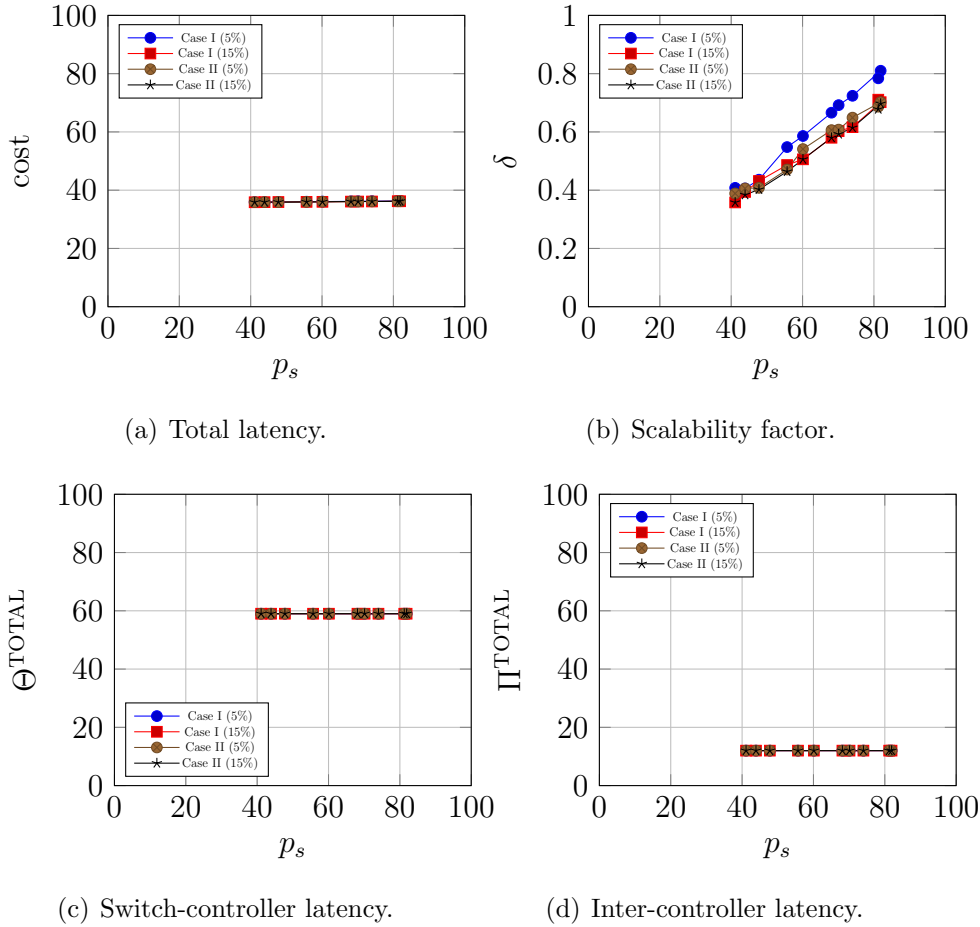


**Figure 6.3:** Results for Topology II, having high connectivity and relatively low number of possible locations for the controllers, using multiple-mapping.

### Topology III:

Results for this topology are shown in Figure 6.4. This topology also presents less links than Topology I, but the number of possible locations for each controller has increased. In this case it is possible to observe that the total latency significantly decreases due to better results for switch-controller latency. Therefore, the model was able to compensate the reduced number of links, by finding adequate places for controllers that lead to lower latency. Results for all scenarios are the same which indicates that the model is adequate to deal with failure scenarios.

### 6.3 Comparing Multiple Mapping and Single Mapping

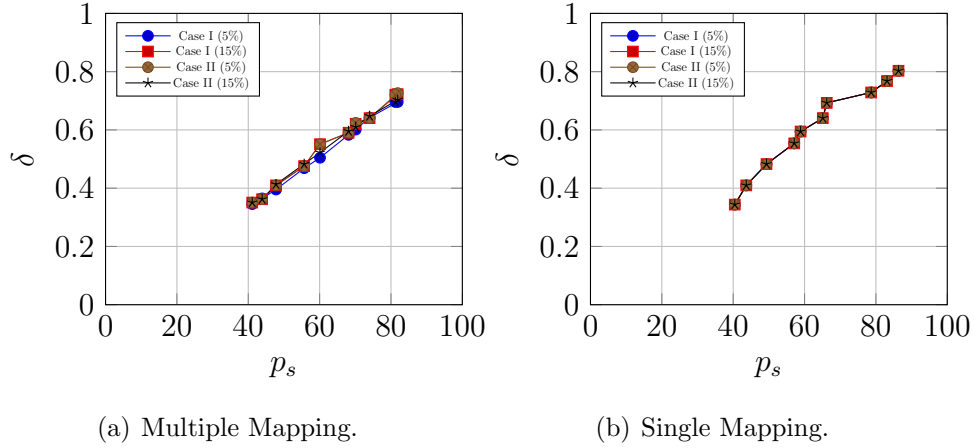


**Figure 6.4:** Results for Topology III, having high connectivity and relatively low number of possible locations for the controllers, using multiple-mapping.

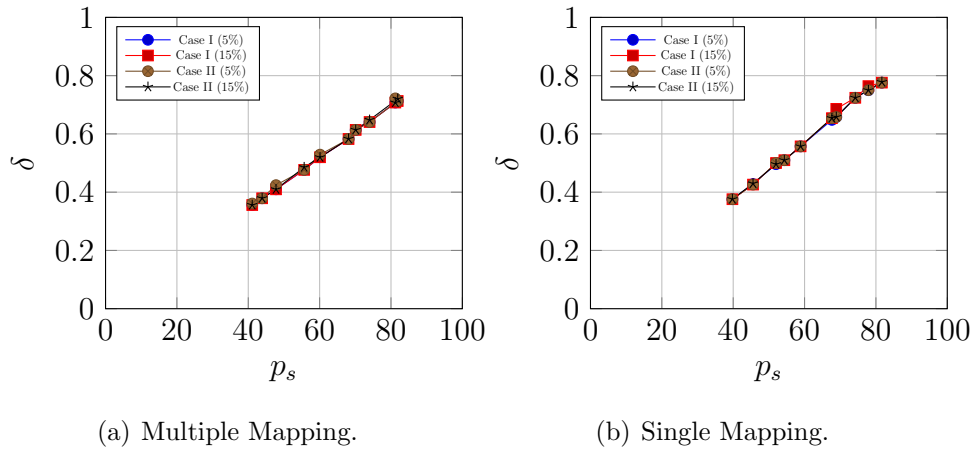
## 6.3 Comparing Multiple Mapping and Single Mapping

In Figures. 6.5, 6.6, 6.7, it is shown that when a single switch is mapped to multiple controllers, the model is able to find the best placement while distributing switches load between controllers, so scalability factor has more freedom since there is less chance to have overloaded controllers. As a result, the overall performance of the network is optimized.

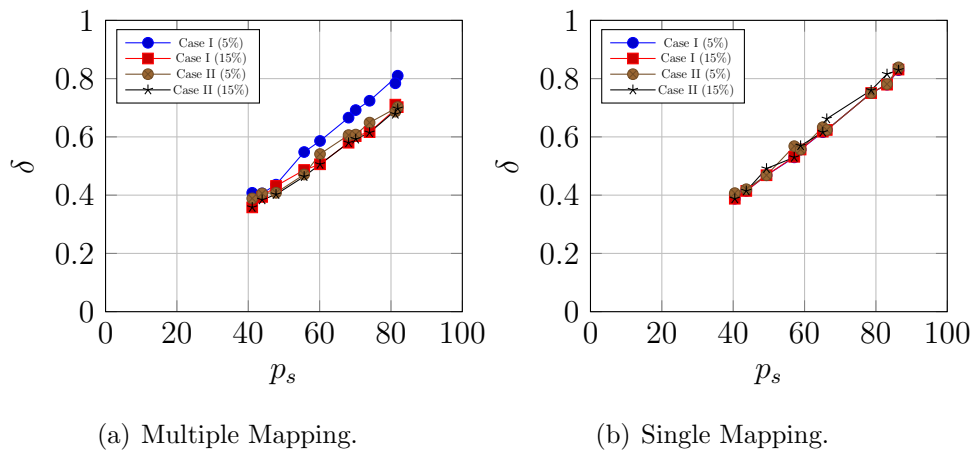
### 6.3 Comparing Multiple Mapping and Single Mapping



**Figure 6.5:** Scalability factor for Topology I, having high connectivity and relatively low number of possible locations for the controllers.



**Figure 6.6:** Scalability factor for Topology II, having low connectivity and relatively low number of possible locations for the controllers.



**Figure 6.7:** Scalability factor for Topology III, having low connectivity and relatively high number of possible locations for the controllers.

## 6.4 Conclusions

The results show that scalability is ensured under different failure scenarios, while latency increase can be compensated through more freedom in controller's locations. The model also serves adequately multiple failure scenarios, presenting similar results for more critical failure scenarios and less critical ones.

In general, in the multiple mapping case the results show that the model is able to keep scalability ( $\delta$ ) better than single mapping case while considering failure scenarios, ensuring load balancing among controllers. The latency may increase when less network connectivity decreases, but this might be avoided if more possible locations for controllers are allowed. Results are similar for Cases Ia, Ib, IIa and IIb, meaning that the model makes a controller placement that serves adequately multiple failure scenarios.

---

## Conclusions and Future Work

---

Programmable networks, like SDN, are getting more and more attention over the last years. In such kind of architectures, the controllers are the most important elements, and most of the ongoing research focus on different issues about them. One of the most important research topic has to do with finding the best placement for controller, while fulfilling different criteria like latency and/or failover.

In this thesis, a scalable and reliable model for controller placement is introduced. This model is mathematically formulated and optimal solutions for controller placement, under different failure scenarios, are obtained. This study tries to find for the best placement while considering failover and scalability. A first mathematical model regarding single mapping scenarios has been implemented, and results have been obtained and analyzed in chapter 5. Results indicate that scalability is ensured under different failure scenarios while resilience is guaranteed. The second mathematical model regarding multiple mapping is also implemented, and results overcome the ones obtained for single mapping due to the flexible nature of multiple mapping approaches.

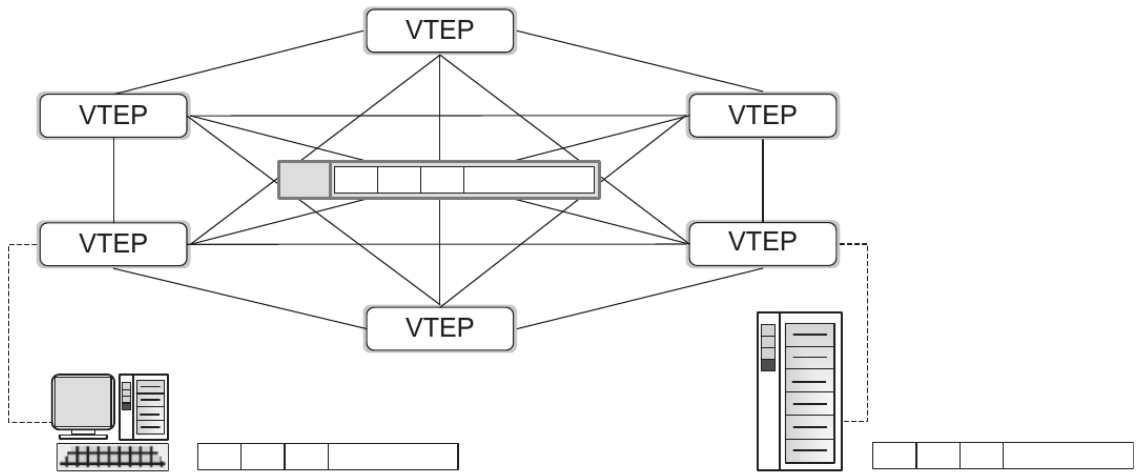
Recently, switches implemented as software applications, usually running in conjunction with a hypervisor in a data center rack, have been proposed [26]. Like virtual machines, virtual switches may be instantiated or moved under software control. Therefore, given an existing physical network, the placement of controllers can be planned having switch virtualization into consideration. Such scenarios are to be considered in future work. An issue that must be considered is that when there are multiple controllers, state synchronization is required to keep network information consistent among controllers. One might think that full inter-controller connectivity is the best choice, as in [13], but this brings scaling problems. That is, state synchronization load, to maintain a consistent information state among controllers, should be reduced for scalability [28]. In [27] it is stated that the controller load can be reduced, and load balance among neighboring controllers is achieved, if controller needs to communicate only with its local neighbors, within a predefined number of hops. Having this issue in mind, the approach can be to provide a virtual

## Conclusions and Future Work

---

controller topology (expected connectivity between controllers) for further embedding at the physical topology. This way any inter-controller connectivity can be provided as input. Such approach, together with virtual switch instantiation, will allow for physical resources to be efficiently utilized. Figure 7.1 illustrates the use of VXLAN and VTEP technologies to achieve virtual implementation in SDN.

**Figure 7.1:** Hypervisor-Based Overlay Networks.



---

## References

---

- [1] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, “A survey on software defined networking with multiple controllers,” *Journal of Network and Computer Applications*, vol. 103, pp. 101 – 118, 2018.
- [2] P. Goransson and C. Black, “Chapter 2 - why sdn?,” in *Software Defined Networks* (P. Goransson and C. Black, eds.), pp. 21 – 35, Boston: Morgan Kaufmann, 2014.
- [3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM COMPUTER COMMUNICATION REVIEW*, vol. 38, pp. 69–74, APR 2008.
- [4] O. N. Foundation, “Openflow switch specification,” tech. rep., 2011.
- [5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “Nox: Towards an operating system for networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 105–110, July 2008.
- [6] D. Erickson, “The beacon openflow controller,” in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13*, (New York, NY, USA), pp. 13–18, ACM, 2013.
- [7] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On Scalability of Software-Defined Networking,” *IEEE COMMUNICATIONS MAGAZINE*, vol. 51, pp. 136–141, FEB 2013.
- [8] J. Medved, A. Tkacik, R. Varga, and K. Gray, “OpenDaylight: Towards a Model-Driven SDN Controller Architecture,” in *2014 IEEE 15TH INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS, MOBILE AND MULTIMEDIA NETWORKS (WOWMOM)*, IEEE, 2014.
- [9] S. Hassas Yeganeh and Y. Ganjali, “Kandoo: A framework for efficient and scalable offloading of control applications,” in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12*, (New York, NY, USA), pp. 19–24, ACM, 2012.

## References

---

- [10] P. Vizarreta, C. M. Machuca, and W. Kellerer, “Controller Placement Strategies for a Resilient SDN Control Plane,” in *PROCEEDINGS OF 2016 8TH INTERNATIONAL WORKSHOP ON RESILIENT NETWORKS DESIGN AND MODELING (RNDM)* (Jonsson, M and Rak, J and Somani, A and Papadimitriou, D and Vinel, A, ed.), pp. 253–259, 2016.
- [11] G. Yao, J. Bi, Y. Li, and L. Guo, “On the Capacitated Controller Placement Problem in Software Defined Networks,” *IEEE COMMUNICATIONS LETTERS*, vol. 18, pp. 1339–1342, AUG 2014.
- [12] B. Heller, R. Sherwood, and N. McKeown, “The Controller Placement Problem,” *ACM SIGCOMM COMPUTER COMMUNICATION REVIEW*, vol. 42, pp. 473–478, OCT 2012.
- [13] A. Sallahi and M. St-Hilaire, “Optimal Model for the Controller Placement Problem in Software Defined Networks,” *IEEE COMMUNICATIONS LETTERS*, vol. 19, pp. 30–33, JAN 2015.
- [14] A. Sallahi and M. St-Hilaire, “Expansion Model for the Controller Placement Problem in Software Defined Networks,” *IEEE COMMUNICATIONS LETTERS*, vol. 21, pp. 274–277, FEB 2017.
- [15] R. Masoudi and A. Ghaffari, “Software defined networks: A survey,” *Journal of Network and Computer Applications*, vol. 67, pp. 1 – 25, 2016.
- [16] R. El-Azouzi, D. Menasche, E. Sabir, F. De Pellegrini, and M. Benjillali, *Advances in Ubiquitous Networking 2: Proceedings of the UNet16*. Lecture Notes in Electrical Engineering, Springer Singapore, 2016.
- [17] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” *PROCEEDINGS OF THE IEEE*, vol. 103, pp. 14–76, JAN 2015.
- [18] F. Bannour, S. Souihi, and A. Mellouk, “Scalability and reliability aware sdn controller placement strategies,” in *2017 13th International Conference on Network and Service Management (CNSM)*, pp. 1–4, Nov 2017.
- [19] H. Yannan, W. Wendong, G. Xiangyang, Q. Xirong, and C. Shiduan, “On Reliability-optimized Controller Placement for Software-Defined Networks,” *CHINA COMMUNICATIONS*, vol. 11, pp. 38–54, FEB 2014.
- [20] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, “Controller Placement and Flow based Dynamic Management Problem towards SDN,” in *2015 IEEE*

## References

---

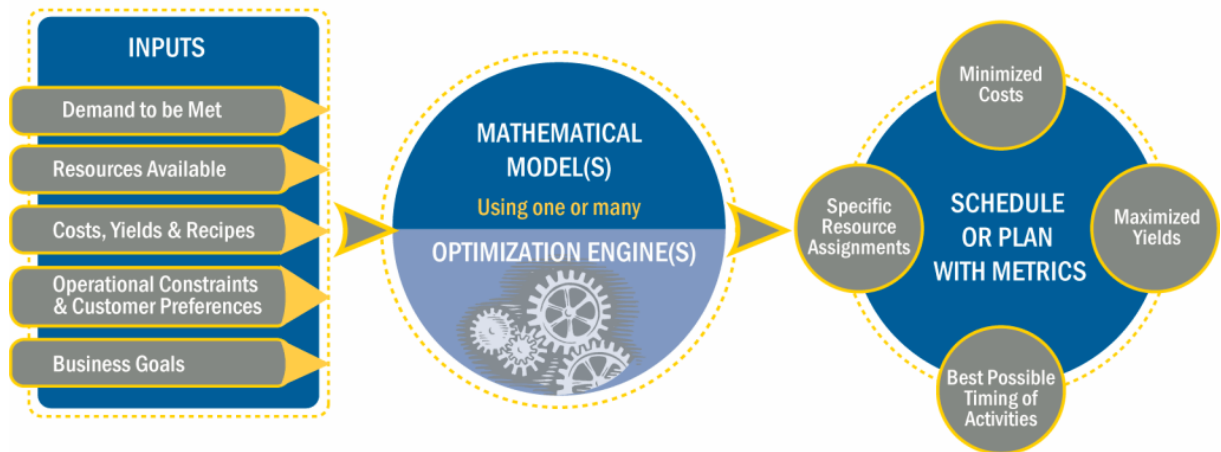
- INTERNATIONAL CONFERENCE ON COMMUNICATION WORKSHOP (ICCW)*, IEEE International Conference on Communications Workshops, pp. 363–368, IEEE, 2015.
- [21] G. Cheng and H. Chen, “Game model for switch migrations in software-defined network,” *ELECTRONICS LETTERS*, vol. 50, pp. 1699–1700, NOV 6 2014.
- [22] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, “Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks,” *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, vol. 12, pp. 4–17, MAR 2015.
- [23] T. Y. Cheng, M. Wang, and X. Jia, “QoS-Guaranteed Controller Placement in SDN,” in *2015 IEEE GLOBAL COMMUNICATIONS CONFERENCE (GLOBECOM)*, IEEE Global Communications Conference, 2015.
- [24] B. P. R. Killi and S. V. Rao, “Controller placement with planning for failures in software defined networks,” in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, Nov 2016.
- [25] A. Sallahi, “Optimal placement of controllers in software defined networks,” mthesis, Department of Systems and Computer Engineering, Carleton University, May 2014.
- [26] P. Gransson, C. Black, and T. Culver, “Chapter 8 - sdn in the data center,” in *Software Defined Networks (Second Edition)* (P. Gransson, C. Black, and T. Culver, eds.), pp. 191 – 215, Boston: Morgan Kaufmann, second edition ed., 2017.
- [27] S. Schmid and J. Suomela, “Exploiting locality in distributed sdn control,” in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13*, (New York, NY, USA), pp. 121–126, ACM, 2013.
- [28] Z. Guo, M. Su, Y. Xu, Z. Duan, L. Wang, S. Hui, and H. J. Chao, “Improving the performance of load balancing in software-defined networks through load variance-based synchronization,” *Computer Networks*, vol. 68, pp. 95 – 109, 2014. Communications and Networking in the Cloud.

## Implementation

### A.1 Introduction

The general approach used by CPLEX to solve mathematical models is shown in Figure. A.1. In this thesis, additional steps were required for tests to be performed, whose flowchart is shown in Figure. A.2. It starts with a set of initial inputs which include the existing nodes and switches, number of controllers, together with physical links. A python algorithm builds failure scenarios and adds the corresponding constraints to the base model.

**Figure A.1:** General structure of CPLEX solving model.  
The Structure of Optimization Models



The data model is applied to CPLEX so that the best location for facilities is found by CPLEX solver. Such solver is able to find the global optimal solution.

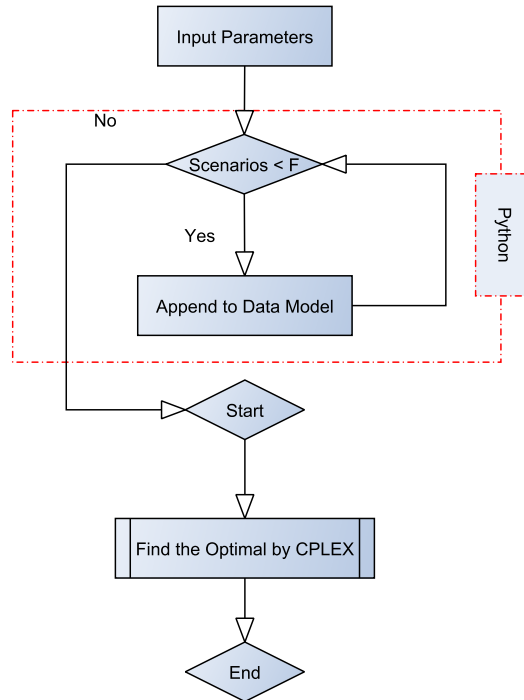
### A.2 Building the Model

To build an optimization problem for CPLEX, we need to define 2 different objects:

## A.2 Building the Model

---

**Figure A.2:** Main flow chart of a test (F is the number of scenarios).



- Modeling objects for defining the optimization problem (constraints, objective function, etc.) They are grouped into an IloModel object representing the complete optimization problem.
- Solving objects for problems represented by modeling objects to be solved. A solving object reads a model, extracts its data, solves the problem and answers queries on solution.

In the following, it is shown the modeling object and solving object, respectively.

```
/*  
*****  
* OPL 12.8.0.0 Model  
* Author: Mohammad Ashrafi  
* Creation Date: Apr 19, 2018 at 4:34:04 PM  
*****/  
  
int nSwitches =...;  
int nLinks =...;  
int nControllers =...;  
int nNodes =...;  
int Number_of_scenario=...;
```

## A.2 Building the Model

---

```
float k=50;
float DL=100;
range nodes=0..nNodes-1;
range scenarios=0..Number_of_scenario-1;

tuple edge{

    int scenario;
    int source;
    int dest;
}
tuple switch{
    int loc;
    float load;
}
tuple controller{
    int id;
    float capacity;
}

{int} possible_node= ...;
{edge} links=...;
{switch} switches=...;
{controller} controllers=...;

dvar boolean sigma[controllers][nodes];
dvar boolean mu[controllers][controllers][possible_node] [
    ↪ possible_node][links];
dvar boolean beta[links];
dvar boolean gamma[switches][controllers][scenarios];
dvar boolean phi[switches][controllers][links][possible_node] [
    ↪ scenarios];
dvar boolean lambda [switches][controllers][possible_node][scenarios
    ↪ ];
dvar float+ delta ;
dvar float+ PI;
dvar float+ THETA;
dexpr float cost = delta+k*THETA/DL+k*PI/DL;
```

## A.2 Building the Model

---

```
minimize cost ;

subject to {

//formula (2)
forall(c in controllers)
node_unique:
sum(n in possible_node)sigma[c][n]==1;

forall(n in possible_node)
node_1:
sum(c in controllers)sigma[c][n]<=1;
//formula (3)
forall(s in scenarios,ci in controllers, cj in controllers,ni in
    ↪ possible_node,nj in possible_node:
ci.id!=cj.id && ni!=nj )
{

conservation_law_src:
sum(l in links:l.source == ni && l.scenario==s) mu[ci][cj][ni][nj][l]
    ↪ == sigma[ci][ni] ;
}

forall(s in scenarios,ci in controllers, cj in controllers,ni in
    ↪ possible_node,nj in possible_node:
ci.id!=cj.id && ni!=nj )
{

conservation_law_dst:
sum(l in links:l.dest == nj && l.scenario==s) mu[ci][cj][ni][nj][l]
    ↪ == sigma[cj][nj] ;
}

forall(s in scenarios,n in nodes, ci in controllers, cj in
    ↪ controllers,ni in possible_node,nj
in possible_node: ci.id!=cj.id && n!=nj && n!=ni )
{

conservation_law_int:
sum(l in links:l.source == n && l.scenario==s) mu[ci][cj][ni][nj][l]
```

## A.2 Building the Model

---

```
    ↪ -
sum(l in links:l.dest == n && l.scenario==s ) mu[ci][cj][ni][nj][l]
    ↪ == 0 ;
}

//formula (4)
forall(s in scenarios,ci in controllers,ni in possible_node,nj in
    ↪ possible_node,cj
    in controllers,l in links:ci!=cj && s==l.scenario)
    mu[ci][cj][ni][nj][l] <= beta[l];

//formula (5)
/*forall(s in scenarios)
loadl_flow:
sum(l in links: s==l.scenario) beta[l] * 0.5 <= PI;*/
//formula (5)
loadl_flow:
sum(s in scenarios,l in links: s==l.scenario) beta[l] * 0.5 == PI;

//formula (6)
forall(s in switches,sc in scenarios)
switch_flow:
sum(c in controllers) gamma[s][c][sc] == 1;

//formula (7)
forall(c in controllers,sc in scenarios)
controller_load:
sum(s in switches) gamma[s][c][sc] * s.load <= c.capacity * delta ;

//formula (8)
forall(s in switches,n in possible_node,c in controllers,f in
    ↪ scenarios)
formula8:
    lambda[s][c][n][f]>=gamma[s][c][f]+sigma[c][n]-1;

//formula 9
forall(sc in scenarios,nj in possible_node,s in switches, c in
    ↪ controllers)
```

## A.2 Building the Model

---

```
{

Switch_controller_latency_src:
sum(l in links:l.source == s.loc && sc==l.scenario ) phi[s][c][l][nj
  ↪ ][sc] ==
    lambda[s][c][nj][sc] ;

}

forall(sc in scenarios,nj in possible_node,s in switches, c in
  ↪ controllers)
{
Switch_controller_latency_dst:
    sum(l in links:l.dest == nj && sc==l.scenario) phi[s][c][l][
  ↪ nj][sc] ==
    lambda[s][c][nj][sc] ;

}

forall(sc in scenarios,n in nodes,nj in possible_node,s in switches,
  ↪ c in controllers:n!=nj
&& n!=s.loc )
{
Switch_controller_latency_int:
sum(l in links:l.source == n && sc==l.scenario) phi[s][c][l][nj][sc]
  ↪ -
sum(l in links:l.dest == n && sc==l.scenario) phi[s][c][l][nj][sc] ==
  ↪ 0 ;

}

//formula (10
/*forall(c in controllers,s in switches,sc in scenarios)
highest_latency:
sum(l in links:sc==l.scenario,ni in possible_node) phi[s][c][l][ni][
  ↪ sc] <= THETA;*/

// formula (10
sum(sc in scenarios,c in controllers,s in switches,ni in
  ↪ possible_node,l in links:sc==l.scenario)
phi[s][c][l][ni][sc] == THETA;
```

## A.2 Building the Model

---

```
}

execute
{
  var f=new IloOplOutputFile("output.txt");
  f.writeln("[vars]");
  f.writeln("Number of Nodes = ",nNodes);
  f.writeln("Number of controllers = ",nControllers);
  f.writeln("Number of Switches = ",nSwitches);
  f.writeln("");
  f.writeln("cost = ",cost);
  f.writeln("sigma = ",sigma);
  f.writeln("gamma = ",gamma);
  f.writeln("switches = ",switches);
  f.writeln("Theta = ",THETA);
  f.writeln("Delta = ",delta);
  f.writeln("PI = ",PI);
  f.writeln("controllers= ",controllers);

  f.close();
}
```