Delft University of Technology
Master of Science Thesis in Computer Engineering

Early-Warning-Driven Approach for Network Protection Against Earthquakes

Nikoleta Zaharieva

Embedded
Networked
Systems

TU Delft Delft University of Technology

Early-Warning-Driven Approach for Network Protection Against Earthquakes

Master of Science Thesis in Computer Engineering

Embedded and Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Nikoleta Zaharieva
n.z.zaharieva@student.tudelft.nl
nikoleta.zaharieva@gmail.com

8th February 2022

Author
Nikoleta Zaharieva (n.z.zaharieva@student.tudelft.nl)
(nikoleta.zaharieva@gmail.com)

Title
Early-Warning-Driven Approach for Network Protection Against Earthquakes

MSc Presentation Date
22nd February 2022

Graduation Committee

Prof. dr. ir. Fernando Kuipers	Delft University of Technology
Dr. Kaitai Liang	Delft University of Technology
Ir. Jorik Oostenbrink	Delft University of Technology

Abstract

Natural disasters can significantly disturb communication networks. There are examples of events causing massive connectivity failures in the past, such as the Great East Japan Earthquake. Network protection mechanisms have been developed to cope with the destructive power of natural disasters and mitigate their impact on connections availability, but in terms of accuracy, they are far from perfect.

Some natural disasters are predictable and can be detected hours or even days in advance. In that case, an adequate protection strategy can be applied. Nevertheless, other types of disasters, such as earthquakes, are classified as unpredictable; thus, protecting the network becomes challenging. Fortunately, early warning systems can detect ground motion and provide a few seconds of warning before the shaking is actually felt. In our work, we utilize early warnings and other disaster data to develop a network protection approach against earthquakes, which operates under rigorous time constraints. Our goal is to minimize the number of disrupted connections in the network by rerouting as many connections as possible out of the disaster zone, such that their availability is maximized. At the same time, the sum of the bandwidth of the connections in the network is also maximized.

We create a realistic disaster model using an early warning system and disaster information. We tackle the uncertainties related to unpredictable disasters by introducing the concept of multiple disaster scenarios. We define the problem of finding paths with maximized availability considering the multiple scenarios. The problem is extended further by adding bandwidth constraints. We propose heuristics to solve the formulated problems and provide an SDN implementation. We validate the effectiveness of our solutions by conducting a series of experiments and creating a custom metric to evaluate our results. The results show that our approach improves the availability of the endangered connections; using the proposed multi-scenario strategy is more beneficial than a single scenario. The results also show that our bandwidth algorithm can optimize the bandwidth utilization of the network. Finally, we compare our solution to an exact solution and find out that our results are very close to optimal. This work provides a mechanism for network operators to ensure the protection of critical network communication in the event of a natural disaster and prevent the potential loss of human lives.

Preface

This report contains my thesis project- the final step for obtaining a Master of Science degree in Computer Engineering and marks the end of my journey at the Delft University of Technology. During the time spent on this project, I gained in-depth knowledge of communication networks and network protection strategies. Moreover, I realized how much society is dependent on reliable connectivity and how severely a natural disaster can damage network connections. Even though it is beyond our capabilities to control the occurrence of disasters, it is substantial to find a way to manage their outcome.

First of all, I would like to thank Professor Fernando Kuipers for giving me the opportunity to work on this project within the Embedded Networked Systems research group. I sincerely appreciate the guidance and the valuable insights he gave me throughout the project. Also, I would like to express my sincere gratitude to my daily supervisor, Jorik Oostenbrink, for his detailed feedback, constructive criticism, and the time and patience along the way. Furthermore, I want to thank Dr. Kaitai Liang for being a thesis committee member.

I am grateful that at TU Delft, I met people who became my friends and would like to thank them for being part of my Delft experience. To my friends back home and here in the Netherlands- having you around (despite online) in these extraordinary pandemic times and less than favorable circumstances is invaluable. Special thanks to Stefan for being supportive and encouraging and keeping me motivated all the time.

Finally, I would like to express my deepest gratitude to my parents for their unconditional support. I would have never accomplished this without them. Last but not least, I thank my sister for always believing in me and pushing me to step outside of my comfort zone.

Nikoleta Zaharieva

Delft, The Netherlands
13th February 2022

Contents

Chapter 1

Introduction

Data communication has a fundamental role in multiple aspects of today's life. With the rapid development of technological innovations, network reliability is no longer critical only in social communication between people. A network failure can lead to the interruption of critical services resulting in severe financial losses. Highly essential industries like transportation, medicine, and energy supply depend on reliable network connectivity. Ensuring end-to-end connectivity can be crucial, especially in emergencies, when people have to be evacuated or need to get in contact with emergency operators. For this reason, network resilience has been a research topic for the past decades.

Network failures can occur for multiple reasons - single component failure, targeted human-caused attacks, or a natural disaster such as earthquakes, floods, hurricanes, and tsunamis. Specifically, natural disasters are large-scale, which puts at risk entire geographical regions to be disconnected from the rest of the world. Unfortunately, there are examples of such events - The Great East Japan Earthquake in 2011; hurricane Katrina in 2005 [12] among others.

Protecting the network infrastructure from the destructive power of natural disasters is, of course, a challenging task. Hence, network protection mechanisms have been developed to tackle this problem. There are proactive(protection) approaches that aim to mitigate the damage before the event occurs and reactive(recovery) approaches that handle the post-disaster situation. The former has the benefit of being fast but not always successful, whereas the latter might be time-consuming but more resource-efficient [17]. Depending on the scale of the impact, network recovery operations might take hours, days, or even months before the connectivity is completely restored. In the current work, we develop a proactive protection strategy. However, in contrast to the common protection strategies, ours is initiated after the disaster has occurred but prior to damaging the network components.

1.1 Problem Statement

Protecting a network from a natural disaster is complex due to two main factors: the scale of the disaster and the uncertainty associated with its occurrence. It is impossible to predict the exact course of every possible disaster that might strike a network. Fortunately, there are early-warning systems that could provide

information about an upcoming earthquake in advance by detecting ground shaking [28]. This way, we can protect the network's sections where damage is expected by redirecting traffic according to the received information.

However, the warning is received shortly before the disaster strikes and the time available for action is strictly limited. Considering the large scale and the severe time constraint, it would be impossible for a network operator to deal with this task; thus, it would be ideal if the network could protect itself autonomously.

To our knowledge, the existing work that addresses the problem of protecting the existing traffic in networks does not address the uncertainty of natural disasters sufficiently. They usually consider a particular scenario and extensive knowledge of its impact on the network. Therefore, we propose a system that receives and responds to an early-warning input and takes autonomous and dynamic re-routing decisions to find an alternative path with maximized availability. We address the uncertainty related to the lack of knowledge of the exact characteristics of the disaster and the damage it will cause by incorporating multiple scenarios.

1.2 Goal and Contributions

This work aims to mitigate the impact of the disaster by minimizing the number of disrupted connections, maximizing the availability of the connections, and satisfying their requested bandwidth.

Our contributions are as follows:

- We assume the presence of an early warning system to provide disaster information, and based on that, we generate a set of multiple potential disasters. Under each scenario, network components fail with a certain probability, depending on their proximity to the epicenter and the disaster's impact.
- We propose a heuristic approach for finding an alternative path with minimal failure probability considering multiple scenarios.
- We provide a comprehensive disaster model and implement the proposed approach in SDN. We show experimental results to demonstrate the timing performance of the algorithm and its capability to handle large-scale failures within time constraints. Our results validate that our multi-scenario approach outperforms the usage of a single scenario. We prove that our proposed heuristic achieves results comparable to an exact solution but within less time.
- We propose a new algorithm to optimize bandwidth utilization in a network.

1.3 Thesis Outline

The organization of this thesis is the following:

- Chapter 2: Background - provides background information about concepts and topics used further in this report

- Chapter 3: Related work - provides an overview of the existing related research
- Chapter 4: Approach - describes a proposed algorithm to create a self-protecting network and an extension to this approach that provides bandwidth guarantees
- Chapter 5: Experiments - describes the experimental setup, disaster model, numerical results showing the performance of the proposed approach, and analysis of the achieved results
- Chapter 6: Conclusion - summarizes the thesis and presents some conclusions
- Chapter 7: Future work - finalizes this thesis and gives directions for further research and areas that can be improved

Chapter 2

Background

This chapter comprises a brief overview of relevant topics required for a complete understanding of the remaining chapters of this work. We present the concept of Software-defined networking and its advantages over traditional networking and argue why it is suitable for implementing a self-protecting network. Next, we discuss the OpenFlow protocol, followed by an introduction to disaster Early Warning Systems and earthquakes.

2.1 Software-defined networking and OpenFlow protocol

Traditional networks consist of physical devices with dedicated functions - routers, switches, firewalls, etc. Routers constantly exchange routing information to obtain an overview of the network. A significant drawback of these conventional networks is that they lack flexibility - the initial configuration and any configuration change of every device have to be performed by the network operator. Thus, the complexity scales as the network size grows. Apart from this, different vendors usually supply different hardware devices which run proprietary software, making the management and maintenance of the network even more troublesome.

In conventional networking, each component has a combined control and data (forwarding) plane. The control plane handles the device configuration and routing information and, based on that, packet forwarding is done in the data plane. Once the forwarding information is sent to the data plane, it becomes difficult to change, which contributes to an increase in the network management complexity [36].

The concept of Software-defined networking was developed to address and overcome these burdens. The main feature of SDN is the decoupling of the control and data plane. The control function is transformed from distributed (in traditional networks) to centralized. The role of a decision-making unit is assigned to a programmable controller with an overview of the entire network. The routing decisions are sent from the controller to the forwarding devices in the data plane using forwarding rules. The difference in the architectures of the traditional network and SDN are presented in Figure 2.1.

The programmability of SDN enables flexible and simplified dynamic traffic

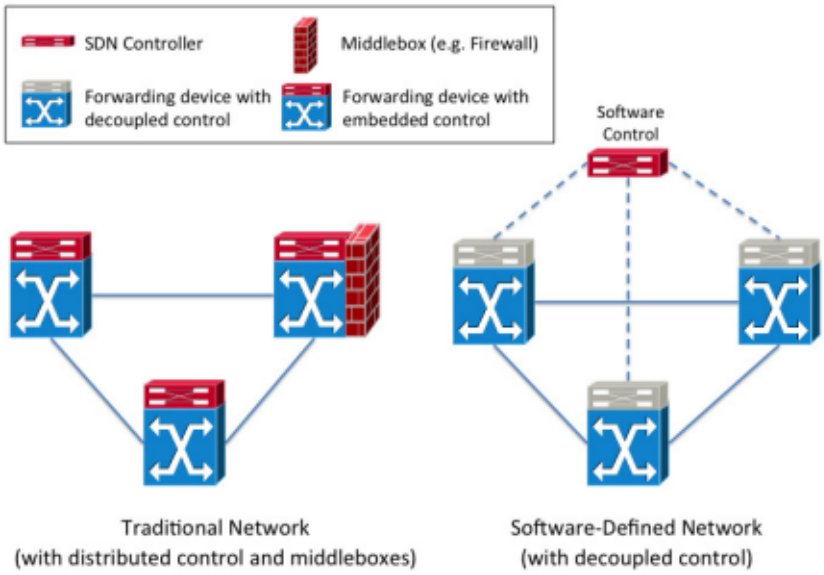


Figure 2.1: **Comparison between traditional network architecture and Software-defined network architecture [27].**

management. The centralized controller with knowledge of the whole network implements customized routing algorithms, and the path computation process is an effortless task. The network behavior is easily adjustable to the ongoing situation in the case of a disaster, as long as the controller is not affected. The centralized overview and the adjustability of the data plane makes SDN a suitable choice for implementing our self-protecting network approach.

The presence of a centralized controller does not bring only benefits. For instance, a challenge could be to eliminate a single point of failure by ensuring redundancy. Another one is choosing appropriate positioning of the controller such that the delay of the switch-controller communication is optimized. However, these issues are not in the scope of this thesis and thus will not be addressed further.

The OpenFlow protocol is one of the standards used for establishing communication between the two decoupled planes, namely the controller and the forwarding devices. The OpenFlow switch architecture is shown in Figure 2.2. The messages exchanged by the controller and the switch determine how different types of packets should be processed [11].

In OpenFlow, the forwarding device contains one or multiple flow tables with flow entries that identify where to forward the packets of any incoming flow. The controller can manage the flow entries proactively or reactively by adding, modifying, or deleting flow entries. The main components of a flow entry are match fields, counters, and instructions. The match field contains metadata, the ingress port, and the packet header, which the incoming packet should match. The matching process begins at the first flow table, following the priority of the flow entry. If an entry that matches the incoming packet is available, the instructions defined in the instruction set are executed. When no match is found

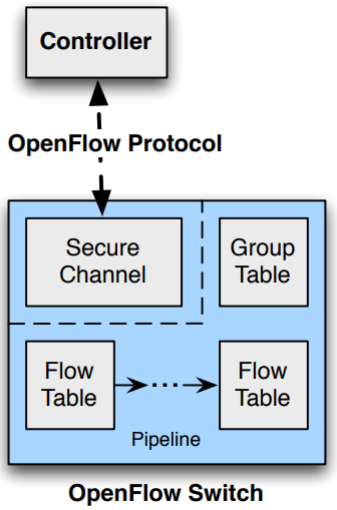


Figure 2.2: **OpenFlow switch architecture [11].**

in the flow table, the instruction from the miss-flow entry is applied.

2.2 Early-warning systems

Early warning systems (EWS) are systems that detect and characterize upcoming natural disasters and provide timely information about the emerging danger[3]. An alarm level is determined to characterize the severity of the danger, and a warning is distributed via a communication system. The warning recipients then take appropriate protective actions. The primary purpose of EWS is to mitigate the destructive effect a disaster might have on a particular infrastructure and evacuate people out of the risk zones on time [25]. There are different early warning systems depending on the type of hazard. In the scope of this work, we are interested only in those hazard events that can cause communication network outages. Examples are geological hazards (earthquake, tsunami, volcanic eruptions, etc.) and Hydro-Meteorological Hazards (floods and storms). The time available for a response, and the action that needs to be taken, are dependent on the type of the event. Sometimes the event can be detected days or hours in advance, but other times only a few minutes or even seconds are available [3].

In the current report, the focus is on earthquake early warning systems (EEWS). However, the approach we propose applies also to other types of early warning systems. The function of EEWS is to rapidly detect and characterize the earthquake and issue an alert/warning to the end-user of a system. Such warnings are widely available in some countries, like Mexico and Japan. In contrast, for instance, in the US and India, only a selection of users receive the warnings [2]. When talking about earthquakes, the available warning time, namely the time frame between detecting the ground shaking and experiencing the shaking by the user, is usually seconds or minutes [2]. Even though the concept of EEWS has existed for over a century, it is still under active develop-

ment to improve the effectiveness and accuracy of the warnings. In [2], three user categories are defined - users who use the warning to make personal decisions, automated response applications, and users who need warning information for situational awareness. In the event of an earthquake, we believe that the impact of potential network failures can be mitigated by creating a self-protecting network mechanism, which falls into the second user category. This mechanism would receive and process a warning from the EEWS and respond automatically by making protective decisions.

2.3 Earthquakes

Earthquakes are natural disasters with devastating impacts - causing human fatalities and economic losses. Apart from the damaging effect of the earthquake itself, often there are hazardous follow-up events such as landslides and tsunamis, which enlarge the damage [5]. The severity of an earthquake can be measured by its characteristics- magnitude and intensity. The magnitude refers to the total radius of the earthquake, and the intensity is a measure of the ground shaking that can be felt at a particular location and is varying throughout the disaster zone [40, 5]. Earthquakes of magnitude lower than 4.5 are unlikely to cause damage. Intensity scales are used to associate an intensity level to the expected level of damage. For instance, a widely-adopted intensity scale is the Modified Mercalli Scale (MMI). The low values in this scale correspond to how shaking is felt by people, and the higher values represent structural damage[41]. Intensity levels below VI usually do not cause physical damage.

Chapter 3

Related Work

Protecting a communication network against large-scale failures due to disasters and mitigating the risk of disrupted connections has been a topic of research in the past decades. This chapter will discuss the existing related work on network protection against natural disasters.

3.1 Protection strategies

The authors of [13] presented an overview of research works addressing network disaster survivability. They discuss different methods to model a disaster and handle its damaging effect. There are three types of approaches to provide connection protection- proactive, reactive, and hybrid. Natural disasters are classified as predictable (when the event can be forecasted in advance) and unpredictable (when a prediction cannot be made). In the scope of this work, we focus only on proactive protection against large-scale failures caused by natural disasters and, more particularly, earthquakes, which are classified as unpredictable natural disasters.

3.1.1 Proactive protection

Izaddoost and Heydari wrote multiple papers on developing a dynamic preventive approach to protect network connections from the destructive behaviour of natural disasters [18], [20], [19], [21]. The authors aimed to improve network survivability by rerouting endangered connections through safer paths with desired availability. They assumed probabilistic component failures that vary in time and depend on the impact radius and disaster intensity. They define a lower and upper path failure probability threshold as a decision parameter for rerouting prior to the failure of the original path. In [18] and [19], Izaddoost and Heydari suggest finding all paths between the source and destination node, filtering out the shortest paths, and evaluating their end-to-end failure probability. The first shortest path is selected if its failure probability is below the lower threshold. If its failure probability is higher than the upper threshold, another shortest path is selected. Even though this method can ensure a safe alternative path, finding all paths for every node pair is computationally intensive; hence this solution will not be feasible for large networks.

Finding all possible paths and comparing them is a brute force approach to finding the optimal solution. We perform such an experiment in our Experiment section and compare it to our proposed approach's performance. The results prove that finding the exact solution is much more time-consuming and does not provide a significantly better solution than our heuristic.

In [20], Izaddoost and Heydari built upon their previous work. They suggested computing the link damage probability at several decision intervals during a disaster and using the average of the maximum and minimum values as a rerouting decision parameter. This parameter is updated dynamically as the disaster area expands. In addition, to computing the link damage probability, they acknowledged the importance of the link in the network by including the betweenness centrality of the link in the computation. In their most recent work [21], Izaddoost and Heydari continued their previous study and proposed a self-adaptive protection model and optimization of the chosen rerouting threshold. The authors used two methods to model the disaster: they used randomly chosen locations in the first method. The second method considered a worst-case scenario by assuming that the disaster can originate from each node. For both methods, no actual seismic or hazard data was used.

In [32] Saito et al. presented a method for disaster avoidance against heavy rainfall by performing network reconfiguration. The proposed approach is implemented in SDN and uses actual geographical information and hazard data from Japan. The system consisted of: disaster risk assessment, disconnection evaluation, and logical network computation. The geographical area is partitioned into grids based on the amount of rainfall, and a warning is issued for each grid at a particular time. After the warning grids are determined, the risk assessment part determines the disaster occurrence probability in this grid. The network reconfiguration aimed to avoid the disaster impact by migrating VMs or switching the path for accessing the VM. The authors considered bandwidth constraints on the communication links and differentiated two types of nodes - access and data center nodes. Their results showed that the execution of the network reconfiguration takes about 100 seconds. The evaluation of this solution is performed on a small network with limited conditions. Therefore the authors extend their work further in [15].

In their follow-up work [15], the authors also considered link transmission delay in addition to bandwidth. They defined three types of nodes- data center, access, and transit nodes. There is a predefined backup location for every data center node where the VM can be migrated. The discussed problem is different from ours because having nodes with specific functions - access and data center nodes- limits the amount of source-destination pairs. In addition, the presence of a predefined set of backup nodes suggests that multiple destinations are possible, similar to [24]. The conclusion from the experiment results was that reconfiguration is more beneficial compared to changing routes. The execution of the proposed algorithm took about 20 minutes. Therefore we can conclude that this approach would not solve our problem, where the response time is within seconds.

In [28] the authors discussed the integration of an early warning system into a network protection strategy. They assumed gathering and processing real-time earthquake information and the presence of a path computation element, which computes and assigns different alarm levels to network components and decides what further action to take. The authors suggested three re-provisioning

approaches - extreme - all connections are re-provisioned with full or degraded bandwidth; relaxed- only critical connections are re-provisioned; careful - some or all connections are re-provisioned depending on the alarm level.

In [16] Iqbal et al. captured the spatiotemporal behavior of natural disasters. They proposed polynomial-time algorithms to detect endangered connections and find a risk-averse path to reroute them under a time constraint. They proposed a grid-based risk model. Every grid cell is assigned with an availability value, which corresponds to the probability of a disaster and this disaster causing damage. The availability of a link is determined by the product of the availabilities of the grids crossed by the link.

3.1.2 Data evacuation

In [44] Xie et al. also discussed the impact of progressive disasters. The authors considered a time-variant network and performed emergency data evacuation. They formulated an optimization to maximize profit and time efficiency. Similarly to us, they assumed that an early-warning system provides helpful information to the operator regarding the time and range of the disaster. The operator can take action accordingly within the available response window. The duration of the warning window is arbitrarily chosen and not based on disaster early warning information. Their simulation used two types of disasters: one that starts from the edge of the network and one from the center. They assumed that at certain time intervals, the disaster expands, and the nodes and links that fall into the disaster area are destroyed; thus, their disaster model is deterministic. Unlike our approach, their work encompasses only predictable disasters. The early warning can be issued days in advance, and the time and impact zone of the disaster can be accurately predicted.

Ferdousi et al. [10] proposed a proactive approach for data evacuation before a disaster impacts the network. Their heuristic, called rapid-data-evacuation, takes the least-delay path to evacuate data from a node located in the disaster zone to a safe node- outside the range of the disaster. They assumed an evacuation deadline and aimed to evacuate the maximum contents within a minimum time. Similar to us, their solution aimed to provide an immediate response based on the received alert. A fundamental difference between this work and ours is that the authors assumed anycast routing from any source to any destination. That was possible because multiple nodes contained a content replica that has to be evacuated. However, that does not apply to our problem because we have no flexibility regarding the source and destination nodes. Another difference is that the authors used a deterministic failure approach and custom disaster model; thus, they assumed complete and specific knowledge on the disaster zone and impact. Since their solution is to find the least-delay path, assuming a probabilistic failure could find a better least-delay path. For the simulation, the authors considered an EMP attack and determined an evacuation window of 7s. However, for EMP attacks, usually, no warning is available.

The authors of [24] also looked into performing data backup prior to a disaster occurrence. Equivalently to us, they assumed the presence of an early-warning system and considered the limited time available for response. The authors aimed to find multiple nodes in a safe zone where data from the endangered node can be transferred. That means that multiple destinations are possible. Thus their approach is less constrained compared to ours. The authors considered

that precisely one node would be affected by the disaster, and respectively, data from only this node has to be migrated. Even though it is not impossible to experience single component failure in real life, we assume such a scenario is not sufficiently representative. Moreover, having only one affected node, while migration to multiple nodes is possible, provides a broader range of nodes where data can be accommodated.

The studies mentioned above discussed proactive protection strategies. Some of them had the same objective as ours- to provide instant action in response to an early warning within a limited amount of time. However, except in [16], the uncertainty of dealing with unpredictable natural disasters is not addressed. The authors assumed detailed knowledge of the disaster specifics and affected components. In contrast, our work does address this issue, and we propose a network protection framework considering multiple disaster scenarios.

3.1.3 Shared Risk Link Groups (SRLG)

The concept of Shared Risk Group (SRG) has been developed to capture the simultaneous failure of network components located in a specific region or multiple regions. SRLGs can be used to model a network failure. However, this approach is usually associated with deterministic failure modeling, where all elements within the SRG fail together [4]. Usually, the common 1+1 protection mechanism ensures full protection, which requires finding two disjoint paths to ensure complete protection. In the context of SRLGs, this will transform into finding an SRLG disjoint path. The SRLG concept is similar to our method - a group of network components fails due to a particular disaster striking. Thus, the multiple disasters can be represented as multiple SRLGs.

Some authors assumed that the failures can have a probabilistic character. The concept of SRG, or SRLG (if only links are considered), evolved into a probabilistic SRLG (pSRLG) to create more accurate models and adapt them to real-world scenarios. Lee et al. [23] proposed a probabilistic SRLG failure model, where the links within an SRLG fail independently with a certain probability. In addition, each pSRLG has its probability. Their solution is categorized as a path protection mechanism. The goal is to determine a pair of primary and backup paths with minimum joint failure probability formulated as an integer non-linear program (INLP). They assumed the SRLG events were mutually exclusive, making their work similar to ours. However, they only consider link failures. Moreover, their approach is not intended to provide emergent network protection.

Diaz et al. [8] based their work on [23], but in addition to risk minimization, they also provided a traffic engineering solution. They proposed a load-balancing approach, finding k pairs of link-disjoint working and protection paths, and selected the best pair in terms of availability.

Yang et al. [45] considered a probabilistic SRLG network to define the problem of availability-based path selection, which is a problem of finding fully or partially disjoint paths of specific availability. They prove that the problem is NP-hard even for a single path in the SRLG network. However, in this work, the SRLGs are not mutually exclusive. In addition, the links within an SRLG fail with probability 1. This work differs from ours first because authors deal with the problem of finding paths to establish connections and not to reroute.

In [30] Pašić et al. present FRAmework for DIsaster Resilience (FRADIR).

The framework includes network design, disaster failure modeling, and protection routing and aims to improve the availability of mission-critical applications. The work is continued further in [31] and [29]. To model the probabilistic regional failures, the authors use Probabilistic SRLGs. The failure probabilities of a network link depend not only on the distance from the disaster's epicenter but also on its availability. A threshold value is defined to determine the list of SRLG. For the network design, the authors incorporate the spine concept. That means that certain edges in the topology are selected and upgraded by increasing their availability. This work differs from ours because the SRLGs are weighted based on the unavailability of the links they are composed of. In this sense, multiple pSRLG is not equivalent to our multiple scenarios- in our case, the probability of a disaster scenario corresponds to its occurrence chance. It is not related to the failure probability of the components affected under this scenario. Moreover, the implementation of the spine concept suggests that this step has to be performed in advance; thus, it cannot be applied as an emergency protection mechanism.

The authors of [22] proposed an algorithm for partially SRLG-disjoint protection. They suggested a classification of the SRLGs based on their failure probability. Two routing models are discussed - Priority Level and Minimum Weighted Risk-disjoint Protection. The authors aimed to present a realistic SRLG assignment; thus, the division of the network into SRLGs is based on seismic hazard maps. As for performance evaluation metrics, they used the number of affected connections and blocked connections. The formation of multiple SRLGs is not related to multiple disasters; hence this work differs from ours.

In [9] the authors considered a risk-aware disaster model by mapping SRLGs to events from a hazard map, which represents the risk events. Dikbiyik et al. proposed traffic engineering solutions for proactive and reactive network protection. Their objective is to minimize the risk and penalty for the network operator in the event of a disaster. The proposed risk model is very generic and includes a broad spectrum of disasters that can occur simultaneously.

In contrast to the above-discussed works, our solution provides a detailed and realistic disaster model and a specific methodology for computing the network component failure probabilities. The component unavailability in our work is based on the component's location regarding the epicenter and the characteristics of the disaster, such as intensity level.

In addition, we assume both links and nodes can fail due to a disaster, which contributes to the completeness of our solution. We manage to handle the uncertainty typical for natural disasters by using multiple disaster scenarios. Even though our multi-scenario strategy is close to the pSRLG concept, we also assume an early warning system, which helps us set boundaries of desired timing performance of our approach. Time limitations are not present in any discussed probabilistic SRLG related work.

Chapter 4

Approach

4.1 Disaster-related uncertainty

As established, natural disasters threaten the connectivity of a communication network. Network components are at risk of failing due to disasters, and ensuring connections will remain undisrupted is a challenging task. While some disasters are predictable and one can perform appropriate protection strategies well in advance, that is not the case when the disaster is unpredictable. Earthquakes are an example of an unpredictable disaster. Hence, protecting a network against an earthquake is possible only after ground motion is detected. In the scope of this project, we will discuss only earthquakes.

In Chapter 2, we discussed that the warning time available for taking preventive action is the time window between the moment the ground shaking is detected and the expected time of the peak ground motion [1]. This time frame is only a couple of seconds, which imposes a rigorous time constraint- observing this limitation, the network connections must be protected. They have to be rerouted away from the hazardous area, if possible. The most time-efficient option is to reroute the connections automatically- without human intervention. For this reason, we aim to create a system that takes as input disaster data and automatically executes a customized rerouting algorithm to protect the connections affected by the disaster.

4.2 Problem Statement

When a disaster strikes, the links in the network, which fall into the disaster zone, are at risk of failure. Thus, their availability and the availability of the paths composed by these links will decrease. In other words, the connections which go through the disaster zone are endangered and have to be protected. We assume that every connection in the network has a pre-defined availability requirement, which corresponds to the end-to-end availability of the connection's path. Hence, in the event of a disaster, the availability of some of the paths will decrease. Meaning the connections associated with those paths will have an increased chance to be disconnected. With this, we come to the problem that these connections will not meet their pre-defined availability requirement. Our objective is to maximize the availability of the connections that do not meet

their availability requirement anymore.

Earthquakes are unpredictable, which means they cannot be detected before ground shaking has started. Therefore, we do not know how the disaster will unfold in advance. We, for example, do not exactly know which links will potentially be damaged and with what probability. If we have information about a historic earthquake, we can compute the availability of the links in the network under the assumption that this particular event will occur again. We can refer to a catalog of historical disasters [42] and observe previous events in a specific geographical area. However, there is still a level of uncertainty about the upcoming disaster, which cannot be captured by only looking into past events. We cannot expect that precisely one of the historical earthquakes will be identical to the current one. Hence, it will be insufficient to take data for only one past disaster and compute component and path failure probabilities. Some authors use hazard maps to identify risk areas and apply a protection mechanism based on that [22], [9], [33]. However, protecting a network using such a generic risk model can come at a high resource cost and still not be effective enough.

As mentioned in Chapter 2, there are early warning systems that can provide a notification when a disaster is detected. However, these systems are not perfectly accurate, especially in a fast-evolving disaster such as an earthquake. Therefore, when receiving a warning message from an EWS, we expect a certain error margin, both in terms of location and magnitude [26], [1]. A way to tackle this possibility of error is to consider multiple disaster scenarios when assessing the components and connections' availability.

If we consider a single disaster scenario where link failures are independent, and set the link weight to be $-log$(1- link failure probability), finding a single alternative path with minimum failure probability is the same as finding the shortest path. This problem is trivial and solvable in polynomial time. However, the failures are no longer independent when multiple scenarios are factored in, making finding a path with minimized failure probability a difficult problem. Finding an alternative path for one connection is a sub-problem of finding alternative paths for all endangered connections. The authors of [23] proved that the problem of minimizing the path failure probability of a single path in probabilistic SRLG networks is NP-complete. This problem is equivalent to our sub-problem, hence our problem is also NP-complete.

To summarize, the problem we have to solve is identifying the endangered connections (connections that do not meet the availability requirement), rerouting them through alternative paths with maximized availability, considering multiple disaster scenarios. In addition, we aim to maximize the sum of the bandwidths of the saved connections while adhering to a capacity constraint. It is important to note that the rerouting operation must be performed under a strict time constraint - within a few seconds after receiving the warning message.

4.2.1 Problem 1: Finding an alternative path with minimized failure probability under multiple scenarios

This subsection will define the problem of finding an alternative path with minimum failure probability for an endangered connection under multiple disaster scenarios.

Prerequisites: We consider a network failure model where multiple disaster scenarios are possible but only one of them will occur. Thus, disaster scenarios are mutually exclusive. Let a disaster scenario be $s \in S$, and the probability of this scenario to occur be π_s. Following the mutual exclusiveness of the disaster scenarios, their occurrence probabilities sum up to 1.

Availability computation: Let $c \in C$ be a connection from the source to the destination node. The availability of a connection is equal to the availability of the path that connects the source and the destination node. In case the component failures are independent, we can compute the availability of a path as the product of the availability of all links and nodes composing this path. The availability (A) of a link (l) or node (v) is the probability that this link or node will not fail and can be computed using (4.1 and 4.2).

$$A_l = 1 - F_l \tag{4.1}$$

$$A_v = 1 - F_v \tag{4.2}$$

Following from this, the availability of the path is computed as shown in (4.3):

$$A_p = \prod_{i=1}^{n} 1 - F_{l_i} \cdot \prod_{i=1}^{n} 1 - F_{v_i} \tag{4.3}$$

Thus, the failure probability of the path is represented as:

$$F_p = 1 - A_p \tag{4.4}$$

Objective: Given the prerequisites above, we can state that our objective is to minimize the path failure probability under each scenario. That can be denoted as follows:

$$min \sum_{s \in S} \pi_s \cdot F_p^s \tag{4.5}$$

4.2.2 Problem 2: Finding an alternative path with minimized failure probability under bandwidth constraint

To make our approach more applicable to real-world situations, we extend the algorithm presented in Section 4.4.1 by adding a bandwidth property. This addition translates to assigning a bandwidth capacity to the links in the network and specifying the amount of bandwidth requested by each connection (source-destination pair). Because in real communication networks, the connections are heterogeneous [34], we assume they request a different amount of bandwidth following certain distribution.

Adding a bandwidth property to the links and the connections in the network imposes an additional constraint to the previously defined problem. That is, the capacity of the alternative path has to comply with the bandwidth requirement of the connections. Thus, we have to extend the problem we defined in Subsection 4.2.1. The essence of the problem remains the same; every connection has an availability requirement. If the current path from source to destination node has lower availability than required, we consider this connection endangered. A new path with a higher availability has to be found. The new problem is

that every link composing the alternative path should have enough capacity to accommodate the incoming connection.

In other words the previously defined *Objective 1* 4.5 remains valid, however under the following constraint:

$$\sum_{c \in C} B_c^l \leq B_{max}^l \tag{4.6}$$

Where B_c^l is the bandwidth of the connections going through the link and B_{max^l} is the maximum capacity of the link.

To evaluate our solution, we have to define a custom compound metric. In this metric, we value availability equal to bandwidth retention. Thus we define our compound metric as $A_p + B$ where A_p is the availability of a connection, and B is the network retention of a connection in percentages.

Objective 2: Our next objective is to maximize the sum over all connections of our custom compound metric.

An alternative path must be computed after establishing that a particular connection needs rerouting. Previously, we used a heuristic to find a path with a maximized availability. However, this is not always as straightforward as before in the current situation. We might encounter a situation where this path does not have sufficient resources- namely, some of the links belonging to the new path do not have enough capacity available (4.6). An option to handle this is to keep looking for another path with enough capacity; otherwise, we have to drop the connection. Unfortunately, we might be unable to find such a path. That is a highly undesirable situation because, in the event of a natural disaster, the traffic going through the network is of great importance and possibly a matter of life or death. Hence, we are motivated to find a solution that avoids dropping connections due to bandwidth constraints while guaranteeing the requested bandwidth and maximized availability. It is crucial to minimize the dropped connections, to maximize the amount of fully functional connections.

4.3 Framework overview

Before going forward with the solution to the problem, we will provide an overview of the system as a whole and the input required to perform the rerouting algorithm. In essence, our system consists of two main components - an operating network and an early warning system. A schematic overview of the entire system is available in Figure. 4.1.

4.3.1 Network input

To protect the network efficiently, we need a centralized, programmable control unit, which receives the warning and controls the routing behavior of all the nodes in the network. While there are various ways to realize this, including traditional networking, we choose to present an SDN implementation. Using a centralized SDN controller to receive the disaster information is practical because not every node has to contain this information.

The controller receives a set of connection requests. We define a *connection* as a pair of source and destination nodes, and all connections have a specified requested availability. The SDN network consists of nodes (switches) V connected to an SDN controller and a set of links L, connecting the nodes. The controller has the role of a path computation unit and runs a customized controller application, where we implemented the rerouting algorithm.

The controller's input is the network topology information — namely, the location of the nodes and the links, characterized by their geographical coordinates. In addition to the source and destination point, each link's geographical information also includes a set of coordinates defining intermediate points, which provides a more accurate view of the actual trajectory of the links. In Chapter 5, we explain in more detail how we make use of this geo-location data.

4.3.2 Disaster Model and Input

The controller receives disaster information containing disaster scenarios and the corresponding links and node failure probabilities. This list is an output created after processing the warning message issued by the EWS. A warning message generated from an EWS contains disaster information depending on the type of disaster. We assume the contents of such messages and extract the information we need to generate disaster scenarios. The proposed self-protecting approach can also apply to other kinds of disasters. However, we will only look into a case study considering earthquakes for this project.

To obtain knowledge about the seismic activity in a particular location, we propose using data from the catalog of past disasters [42] relevant for the specific geographical region where our network is located. We can filter the set of historical disasters by date, magnitude, and intensity. The intensity of an earthquake determines its severity and the level of damage it can cause at a particular location. A widely-adopted intensity scale is the Modified Mercalli (MM) Intensity Scale [41]. Based on this scale, we can conclude that an intensity level lower than VI is unlikely to cause damage to the links in the network. To obtain the size of the expected disaster area and expected intensity level based on magnitude, we use the data provided by the authors of [38] at [37]. Using this data, we can select an earthquake epicenter from the catalog and look at the correlation between disk radius size, the intensity level, and the magnitude of the disaster.

We assume that the early warning message sent from the earthquake EWS provides information about the estimated time of the disaster, the location of the epicenter, and the magnitude of the earthquake. This data is sufficient to form one scenario. However, we assume that this information is not completely accurate, and to cover the possible error margin, we create multiple scenarios. In Chapter 5, we provide a detailed explanation of how we derive the multiple scenarios. By itself, the EWS warning message is insufficient for the controller to take any actions. Therefore, after obtaining the size of the area impacted by the disaster, we can compute failure probabilities for the links and nodes located in those areas. Even though some authors consider only link failures [39], we believe that also including node failures will make our solution more versatile. Thus, our list contains values for both link and node failure probabilities for all scenarios. Using the coordinates of the disaster's epicenter and the radius of the impacted area, network components falling into this area are assigned

a failure probability. The failure probability decreases linearly as the distance from the epicenter increases- so does the intensity level. The component failure probabilities take values between 0 and 1 for intensity levels between VI and X, respectively. If the intensity level is lower than VI, we consider a component does not fail, meaning its failure probability equals 0. To determine the distance of a link from the epicenter, we split the link into line segments based on the provided intermediate points and take the shortest distance from the line segment to the epicenter point.

4.3.3 Prioritizing connections

According to [1], the amount of available warning time at a particular location depends on its distance from the epicenter, which means shorter warning times for connections traversing closer to the epicenter. Therefore, we sort the connections based on their proximity to the epicenter. Connections that traverse closer to the epicenter will be rerouted before connections that are located further away. That is beneficial because connections with a higher chance of being impacted and lower warning times are protected first.

4.4 Proposed Solution

4.4.1 Solution to Problem 1

To solve the problem stated in Subsection 4.2.1, we propose an approach for finding a safer alternative path for connections that are located within the disaster zone. We described the method in Algorithm 1. The used notations are given in Table 4.1.

During the regular operation of the network, before the disaster warning is received, we route the traffic through the shortest path calculated using Dijkstra's algorithm. When a disaster warning is received, we compute component failure probabilities and determine which connections do not meet the availability requirement. The network should be reconfigured to redirect the ongoing traffic through a safer route. We do this by rerouting the traffic outside the disaster zone or through a path with higher availability than the operating path.

To find which connections need to be rerouted, we compute their current availability and compare it to the pre-defined connection availability requirement. If the path availability is lower than the requested value, the connection is marked as endangered, and we need to find a new path. If we find a path that ensures higher availability than the availability of the original path, the flow is redirected to this path. If we cannot find such a path, the original path remains operational. That means the connection is not reroutable, and the traffic flow cannot be protected.

The path with the highest availability can be computed using Dijkstra's weighted shortest path algorithm. If we assign link weights to be the negative logarithmic value of the link availability: $-log(A_l)$, the minimum weight path will be the highest availability path. This solution will give the optimal path if the link failures are independent we fully know the disaster specifics and consider that this particular disaster will occur. However, we are not sure which disaster will occur in our case. We consider multiple disaster scenarios,

each with a respective occurrence probability. This consideration will change the previously discussed way to compute the overall path failure probability in (4.4) because component failures will no longer be independent. To encompass the correlation of the link and node failures for multiple scenarios, the general path failure probability F_g is computed as the sum of the path failure probability for every scenario, multiplied by the occurrence probability π_s of the scenario itself. The proposed solution is a heuristic to find a path with approximately the highest availability. The general path failure probability under all scenarios together is calculated as shown in (4.7):

$$F_g = \sum_{s \in S} \pi_s * F_p^s \tag{4.7}$$

If the operating path has lower availability than requested (rerouting threshold), the system must discover a new, safer path from the source to the destination node. To find such a path, we will use the weighted Dijkstra's algorithm with link weights equal to the negative logarithmic value of the link availability and the node availability(of the tail node): $-log(A_l) + (-log(A_v))$. The difference is that this time, we loop over all scenarios in the list and compute link weights for every link, as shown in Algorithm 2. We exclude the failure probability of source and destination nodes for the availability computation as these nodes are fixedt and cannot be excluded from the alternative path. The algorithm compares the availability of the newly discovered path with the availability of the endangered path. If the availability of the new path is higher, the connection is rerouted; else, the operating path remains unchanged.

The time complexity for the proposed heuristic Algorithm 1 can be denoted as $O(|C|^2 + |C| \cdot (|S||L| + |V|^2)$, where $|C|$ is the number of connections in the network, $|S|$ is the number of scenarios, $|L|$ is the number of links, and $|V|$ is the number of nodes.

Notation	Description
$c \in C$	Connection
$l \in L$	Link
$v \in V$	Node
A_l	Link availability
A_v	Node availability
A_p	Path availability
F_l	Link failure probability
F_v	Node failure probability
F_p	Path failure probability
F_g	General path failure probability under multiple scenarios
π_s	Scenario Probability
F_l^s	Link failure probability under scenario $s \in S$
F_v^s	Node failure probability under scenario $s \in S$
F_p^s	Path failure probability under scenario $s \in S$

Table 4.1: **List of notations.**

Algorithm 1 Finding alternative path for endangered connections

Let C be all the connections in the network
Sort the connections in C ascending on their distance to the epicenter.
for *each connection* $c \in C$ **do**
  $F_g \leftarrow 0$
  Scenarios S are all scenarios
  for *each scenario* $s \in S$ **do**
    $A_p = 1$
    Links L are all the links in connection c
    for *each link* $l \in L$ **do**
      $A_l \leftarrow 1 - F_l$ (Equation 4.1)
      $v \leftarrow$ *tail node of* l
      if $v \neq$ *destination node* **then**
        $A_v \leftarrow 1 - F_v$ (Equation 4.2)
      end if
      $A_p \leftarrow A_p \cdot A_l \cdot A_v$ (Equation 4.3)
    end for
    $F_g \leftarrow F_g + \pi_s \cdot (1 - A_p)$ (Equation 4.7)
  end for
  if $F_g \geq$ *rerouting_threshold* **then**
    Find alternative path using Dijkstra with custom weight function in Algorithm 2
    Compute F_g of alternative path
    if F_g of the alternative path $< F_g$ of original path **then**
      Reroute flows through the new path
    else
      Keep the original path
    end if
  end if
end for

Algorithm 2 Link/Node failure and Link/Node weight computation

$F_l \leftarrow 0$
$F_v \leftarrow 0$ (Destination node)
Scenarios S are all the scenarios
for *each scenario* $s \in S$ **do**
  $F_l \leftarrow F_l + \pi_s \cdot F_l^s$
  $F_v \leftarrow F_v + \pi_s \cdot F_v^s$
end for
if $F_l > 0$ *or* $F_v > 0$ **then**
  $link\_weight \leftarrow -log(1 - F_l) - log(1 - F_v)$
else
  $link\_weight \leftarrow 0$
end if

4.4.2 Solution to Problem 2

To address the problem stated in 4.2.2, we propose an algorithm for finding an alternative path that satisfies the bandwidth requirement. The first step is to observe the link's bandwidth usage - this will give us an overview of how much bandwidth is used and available on each link. To find a higher availability alternative path, we use a modified version of the customized weight function, where at each hop, we evaluate the bandwidth utilization of the link. If the link has insufficient capacity, we mark it as a bottleneck and find a different shortest-path without bottleneck, if possible. A *bottleneck* is a link that does not have sufficient capacity to accommodate the incoming connection. This algorithm is described in Algorithm 4. The notations used are given in Table 4.2.

If it is impossible to find a path without bottlenecks, we use Dijkstra's algorithm again, but this time without considering the bandwidth of the links. In essence, we perform a look-ahead to see which path would have been preferred if there were no bandwidth constraints. Then, we choose this path to be the alternative and try to resolve the previously established bottleneck on this path. To do so, we analyze the utilization of the link and observe whether there is a lower-demand connection, which we can remove and route our connection instead. Note that we remove an ongoing connection only in case:

- *Condition 1:* this would release enough resources to route the incoming connection

and

- *Condition 2:* the incoming connection has higher requested bandwidth

We want to remove only one connection to resolve the bottleneck. If we remove multiple, the number of connections that need to be rerouted will increase. Furthermore, we want to remove a lower-demand connection because otherwise, we will decrease the amount of data transferred through the network, only worsening the situation. In other words, we prioritize connections that require more bandwidth to maximize the network's throughput (see **??**).

To decide which connection to remove, we calculate an optimal bandwidth value using 4.8.

$$B_{opt} = B_h - B_{avail} \tag{4.8}$$

where B_h is the amount of bandwidth requested by the candidate connection and B_{avail} is the residual available bandwidth on the link. Then, we select one connection to be removed from the existing connections going through this bottleneck link. The bandwidth of the selected connection must be higher or equal but closest to the calculated optimal value. After finalizing this step, we iteratively attempt to route the removed connection. We simplify our problem because we are trying to find a path for a connection with a lower bandwidth requirement. As a last resort, it will be dropped if it is not possible to find a safer path to reroute the removed connection.

Encountering a single bottleneck link or, ideally, none is a favorable scenario. However, we can expect that this is not always the case. For this reason, our algorithm should be able to handle situations where the chosen path for rerouting consists of more than one link with insufficient capacity. To handle this

case, we are using multiple layers of recursion. Namely, after resolving the first bottleneck, we go back to the function that finds the shortest path and checks if there is a new obstructed link. This procedure continues until all conflicts are resolved, and the flow can be rerouted successfully to the intended path. The pseudo-code of the algorithm is described in Algorithm 3. The time complexity for Algorithm 3 can be denoted as $O(|C|^2 \cdot (|V|^2 + |C|^2))$, where $|C|$ is the number of connections in the network and $|V|$ is the number of nodes.

Combined with Algorithm 1, the complexity for the full algorithm is $O(|C|^2 + |C| \cdot (|S||L| + |C|^2 \cdot (|V|^2 + |C|^2))$, where Dijkstra's algorithm is replaced by Algorithm 3.

4.4.3 Optimization

For optimization purposes, to avoid spending too much time looking for a path for a particular connection, we set a limit of five attempts. This way, the execution will not be stalled when we encounter a connection requiring an excessive number of attempts to compute a safer path. The need for this optimization is due to the rigorous time constraint.

Notation	Description
$c \in C$	Connection
$h \in H$	Connection that needs to be rerouted
$l \in L$	Link
B_h	Requested connection bandwidth for candidate connection h
B_c	Bandwidth used by existing connection c
B_{max}	Maximum bandwidth capacity of the link
B_{used}	Used bandwidth
B_{avail}	Available bandwidth
B_{opt}	Optimal bandwidth value

Table 4.2: **List of used notations.**

Algorithm 4 Bandwidth extension in custom weight function

if $B_h \leq B^l_{avail}$ **then**
  $weight \leftarrow 1.0$
else if $B_h > B^l_{avail}$ **then**
  Add link to bottlenecks
  No routing through link possible
end if

Algorithm 3 Rerouting with bandwidth constraint

Let H be all the connections that need rerouting
for *each connection* $h \in H$ **do**
  Remove the current bandwidth used by h before finding a new path:
  $B_{used} \leftarrow B_{used} - B_h$
  Find the shortest path and possible bottleneck links with a Weighted Dijkstra's algorithm using a custom weight function (Algorithm 4)
  if path is found **then**
    mark h to be installed
  else if connection h cannot be routed because of limited bandwidth B_{avail} on link l **then**
    Pick the bottleneck link
    $B^l_{avail} \leftarrow B^l_{max} - B^l_{used}$
    $B_{opt} \leftarrow B_h - B^l_{avail}$
    Sort the connections c going through link l ascending on bandwidth usage

    for *each connection* $c \in l$ **do**
      if $B_c < B_h \ \wedge \ B_c \geq B_{opt}$ **then**
        Remove c
        Add c to H
        exit loop
      end if
    end for
    Install h
  else if No path is found **then**
    Drop connection h
  end if
end for

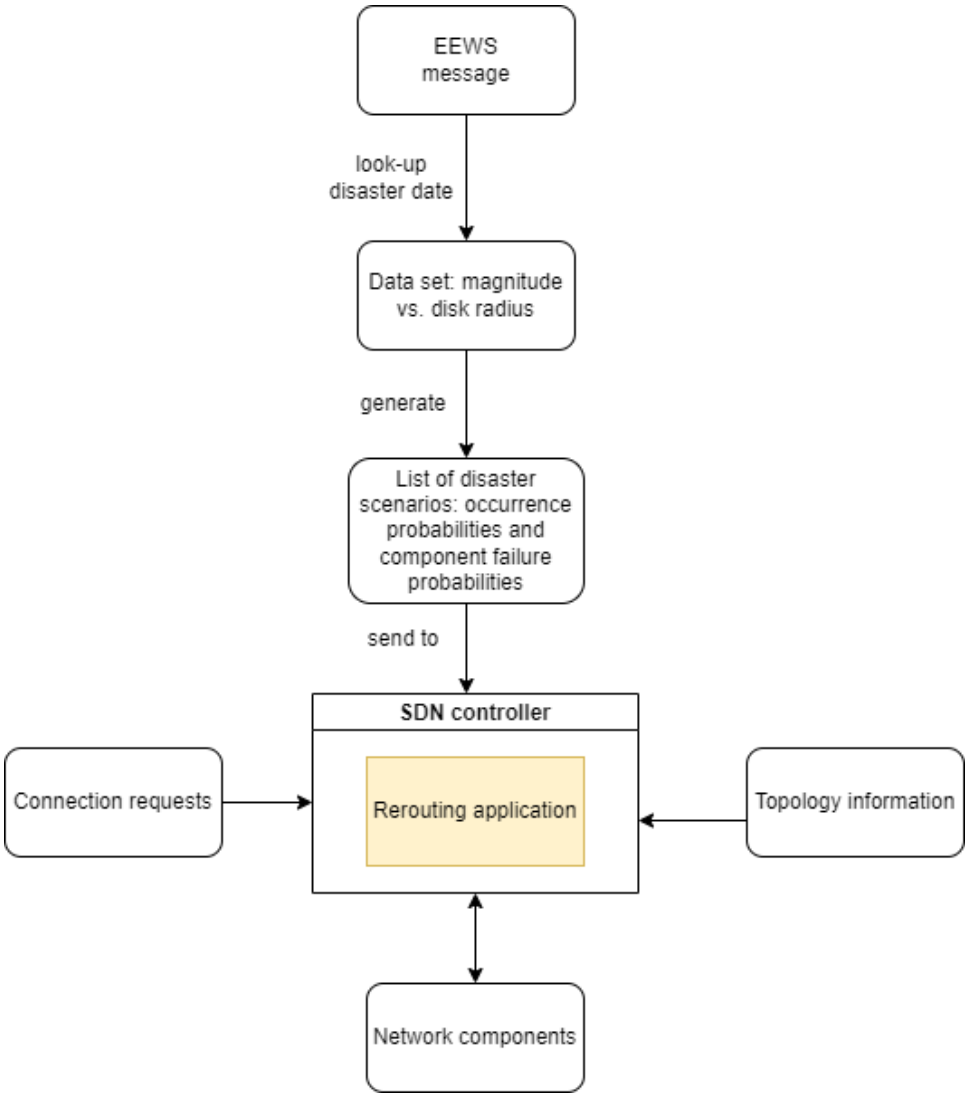


Figure 4.1: **System overview.**

Chapter 5

Experiments

5.1 Setup

For our experiments, we simulated earthquakes located in Italy. We selected three locations for earthquake epicenters - two of them are in places where an earthquake has occurred in the past, according to the USGS catalog of historical disasters [42]. The third epicenter is in a location where the density of the network links and nodes is low. In addition, for each epicenter, we take three additional locations to incorporate the uncertainty of the EWS (described below in Subsection 5.1.1). We use a topology of a network located in Italy, made available by the authors of [43]. The topology and the chosen earthquake epicenters are shown in Figure 5.1. The network consists of 25 nodes and 35 links and is emulated using the Mininet network emulator[7] running on a virtual environment.

5.1.1 Configurations

To model the disaster, we assume that the EWS message provides the expected location and magnitude of the disaster. However, according to [26], it is possible to have an error of ± 1.0 units for the expected magnitude value as well as for the epicenter location. In [1] the authors also discuss error in the magnitude. In our simulation, we address this possible error to tackle this uncertainty and prepare for the upcoming disaster. When we receive a warning indicating a particular magnitude, we generate multiple scenarios to encompass the error by taking additional magnitudes of ± 1.0 units. For instance, if the magnitude value in the warning message is 7.1, we consider it can also be either 6.1 or 8.1 - this corresponds to three possible scenarios for the upcoming disaster. The magnitudes of interest range from 4.6 to 8.1 because an earthquake of a lower magnitude is unlikely to damage a network component. For our experiments, we choose three main magnitude values to cover the entire spectrum.

In addition, we assume that the location information is also not completely accurate. Therefore, we choose three additional location points where the disaster epicenter can be expected for each of the three locations. The extra points are located 20 kilometers from the primary location. To define the coordinates of the three additional points, we set the initial bearing to be 60°, 180°, and 300°, to the main point. This way, we increase the number of scenarios and

Figure 5.1: **Network topology: red asterix- main epicenter location; red cross- additional epicenter locations; blue dot- network node.**

obtain 12 scenarios for each primary location. We assume that the scenarios are mutually exclusive, and each of them has an equal probability of occurring; hence we assign an occurrence probability of 0.083 to each scenario.

We generate a list of node and link failure probabilities for each scenario—the probability for a link or node to fail decreases as the distance from the epicenter increases. However, the component failure probability is not dependent only on its distance from the epicenter; we also consider the properties of the expected disaster. To obtain information about the size of the disaster area, for the different intensity levels and the desired magnitude, we refer to the data set created by the authors of [38]. When we receive the magnitude of the disaster from the EWS, we can use this data set as a reference to find out the corresponding disk size and intensity level.

We assume that the intensity reflects on the failure probability. Therefore we distribute the probability values in the range $[0, 1]$ to decrease linearly for intensity levels between VI and X. Also, the probabilities are distributed linearly by distance; thus, we have an upper and lower bound for each intensity level. For intensity level XI, we assume a constant failure probability of 1 due to the extremely destructive power expected by an earthquake of such a high intensity. The probability range for each intensity level is available in Table. 5.1. In other words, we measure the distance from the links and nodes to the epicenter to determine whether they are located in the disaster area and, if so, in which intensity level and assign a failure probability. To perform more realistic measurements and encompass the curvature of the Earth, we compute the shortest distance between any two points using the *geodesic* module from

the *geopy* Python library [6]. In addition, to find the distance between a link and the epicenter with better precision, we take line segments of the link determined by the available intermediate coordinates and compute the shortest distance from a line segment to the epicenter.

Intensity level	**Lower bound**	**Upper bound**
XI	1	**1**
X	0.8	1
IX	0.6	0.8
VIII	0.4	0.6
VII	0.2	0.4
VI	**0**	0.2

Table 5.1: **Probability ranges.**

All experiments are performed by simulating static traffic consisting of 300 ping requests, combining all possible source-destination pairs. The bandwidth units across the connections are 192, 96, 48, 21, 12, and 3, following the distribution- 1: 2: 4: 10: 10: 20, respectively as proposed in [28]. The capacity of every link is determined by the amount of bandwidth used by the connections going through the link. We performed experiments with three different over-provisioning strategies where the utilization of every link is 40%, 60%, and 80%. In addition, we also performed the experiments without any bandwidth constraint, which is, in essence, the same as having unlimited capacity. The availability requirement of all connections is 0.99.

The experiments are performed in a virtual Ubuntu environment with two virtual cores of an Intel Core i7-8750H Processor (Base 2.20GHz) and 8 Gigabytes of memory.

5.1.2 Objectives

The main objective of our experiments is to observe whether our approach is beneficial for protecting our network. We do this by evaluating if multiple scenarios add a benefit compared to having only one scenario. We also observe the run-time of the proposed algorithm and measure the number of affected connections, how many of them were successfully rerouted and how much is their availability improvement. Another thing we want to look at is the algorithm's behavior as the size of the disaster area increases, and finally, we will look at bandwidth retention after a disaster.

We perform experiments to answer the following questions:

- What is the effect of increasing the magnitude on the number of affected components?
- What is the ratio between the affected and improved connections?
- How does an increase of the disaster area affect the runtime performance?
- What portion of the total time is spent computing, and what portion is spent installing the new path?

- Does the runtime depend on the location of the earthquake?
- Is the achieved response time lower than the time available for reaction imposed by the early warning?
- What is the availability improvement of the newly found paths compared to the original?
- Is there an actual benefit of using multiple scenarios instead of a single scenario?
- How much bandwidth is retained after rerouting the connections?

5.1.3 Evaluation

To evaluate the performance of the proposed heuristic approach, we define metrics for the assessment of the disaster's impact and the effectiveness of the proposed solution. First, we obtain the total amount of affected links and nodes under the considered multiple scenarios. Using this, we derive the number of endangered connections, which is the number of connections eligible for rerouting. To determine which connections have to be rerouted, we set an availability requirement for each connection- if the availability of the path that the connection is currently going through is lower or equal to 99%, it has to be rerouted if possible. We divide the endangered connections into two categories- reroutable and non-reroutable flows. If a connection is reroutable, we can find an alternative path with improved availability compared to the original path. A higher number of reroutable connections means higher effectiveness of the approach.

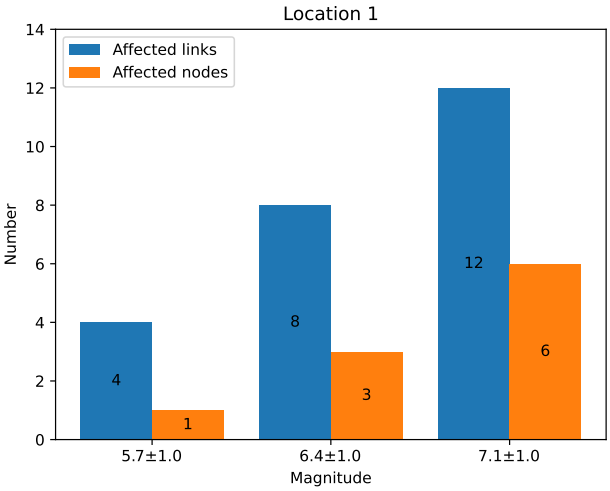
We perform the experiments for multiple magnitude values to test the algorithm's performance when disaster area scales. As the magnitude of the disaster increases, the disaster area expands, and more network components are likely to be affected. We define a node or link as affected if such a component has a failure probability larger than 0. The amount of affected components also depends on the location of the epicenter and the density of the components around this particular location. Therefore, we perform the experiments at three distinct locations. We also performed experiments with multiple and single scenarios where for the case of a single scenario, the occurrence probability is 1.

5.2 Results

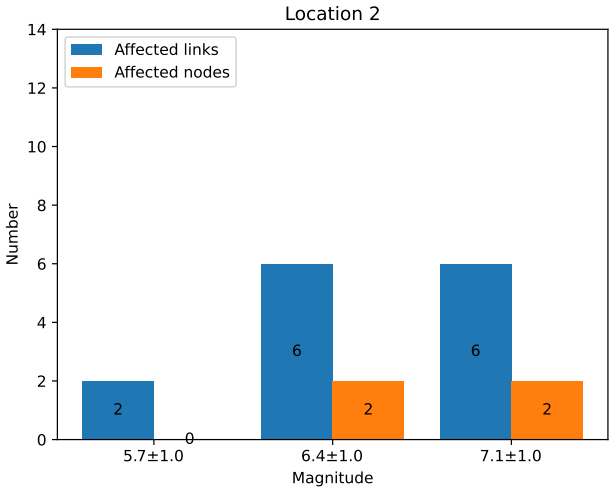
Every experiment is run five times, and the presented results are averaged over the total amount of runs.

5.2.1 Affected components

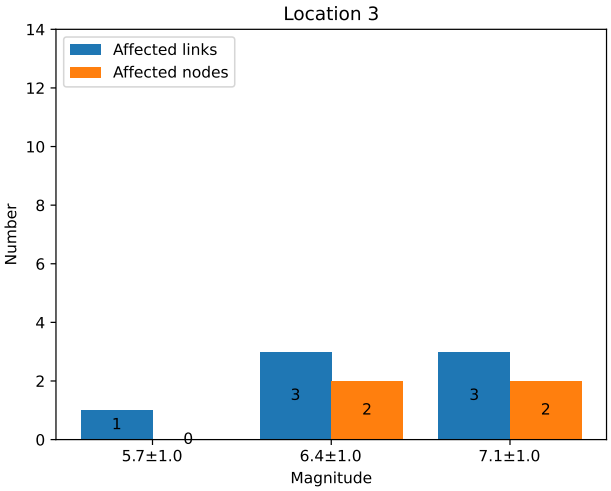
As the magnitude increases, the disaster area gets more extensive, and the amount of affected links and nodes also increases. This trend can be observed for all three locations in Figure 5.2. The most severe increase is visible for location 1, where we see a linear increase in the number of affected links and a superlinear increase in the number of affected nodes. We can explain the difference between the severity of the increase at location 1 and location 2 & 3 by looking at the network's density. Location 1 has more links and nodes in its



(a) Number of affected links and nodes at location 1.



(b) Number of affected links and nodes at location 2.



(c) Number of affected links and nodes at location 3.

Figure 5.2: **Number of affected links and nodes over magnitude for different locations.**

proximity, and that is why more links and nodes are affected when the disaster area increases.

5.2.2 Affected connections

Figure 5.3 shows the total amount of affected connections for an epicenter at location 1 for different magnitude values and different network utilization. We have chosen to showcase location 1 because of its previously mentioned proximity to a dense area. As expected, increasing the magnitude results in more affected connections (non-improved connections + improved connections). That

is strongly related to Figure 5.2, where an increase in magnitude showed an increase in affected network components. Moreover, we can see that the share of improved connections depends on network utilization. The ratio of improved vs. non-improved connections decreases when network utilization increases. The reasoning explains this decrease that when more components in a subset (the disaster area of location 1) of the network fail, the remaining components in that subset reach their capacities by taking overloads of failing components. Nevertheless, even at 80% utilization, our algorithm still saves most of the affected connections.

5.2.3 Time Performance

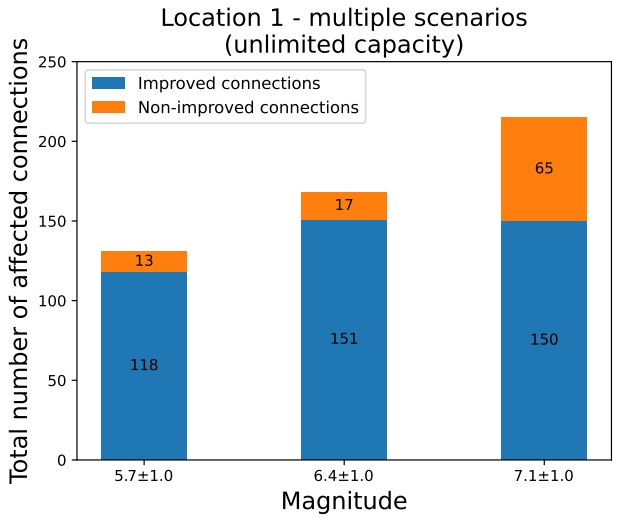
Figure 5.4(a), 5.4(b), and 5.4(c) depict the amount of time spent on computation, installation, and total time, respectively, over increasing magnitude values for all locations. If we look at computation time, Figure 5.4(a), we see an increase in the amount of time when the magnitude of the disaster increases. That is expected because the computation time is closely related to the number of affected components. If we look at the worst-case scenario, we see that location 1 with a magnitude of 7.1 ± 1.0 has a computation time of 25 seconds. That would be above our target, but it is essential to keep in mind that this configuration has a high magnitude and that location 1 is in a highly dense area. We also have an underperforming virtual hardware setup, and with a $2\times$ upgrade of the computation components, our algorithm would be able to handle even our most severe scenario. Finally, let us take notice of locations 2 and 3. We see that lower density areas have significantly lower computation times following our assumption that the computation time is strongly related to the number of affected components. These results show us that the computational intensity of our algorithm is low enough to be used for severe earthquake disasters.

In addition to the computation time, we also measure the time needed to install the flows as shown in Figure 5.4(b). From this figure, we can see a dependency between the installation time and the number of flows that need to be rerouted, as we see a slight increase with the increase in magnitude and slightly higher values for more dense areas.

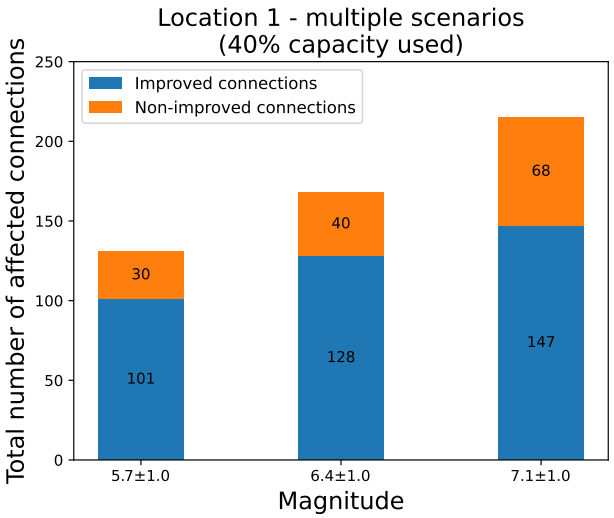
Since the install time is low regardless of the disaster area and epicenter location, we can conclude that the total runtime follows the trend observed at the computation time. Total time for all locations are shown in Figure 5.4(c).

We measured the total time at different locations to see the effect of having multiple against single scenarios on the running time. The results in Figure 5.5 show that having a single scenario gives a lower runtime compared to multiple scenarios, which is expected. The reason is that with multiple scenarios, we have more affected components (see Table 5.2); thus, more connections are considered endangered. Therefore, alternative paths for more connections are being computed and, respectively, installed. However, again we notice that the epicenter location plays a role- in Figure 5.5(b), we see that the difference between single and multiple scenarios is not as significant as in Figure 5.5(a).

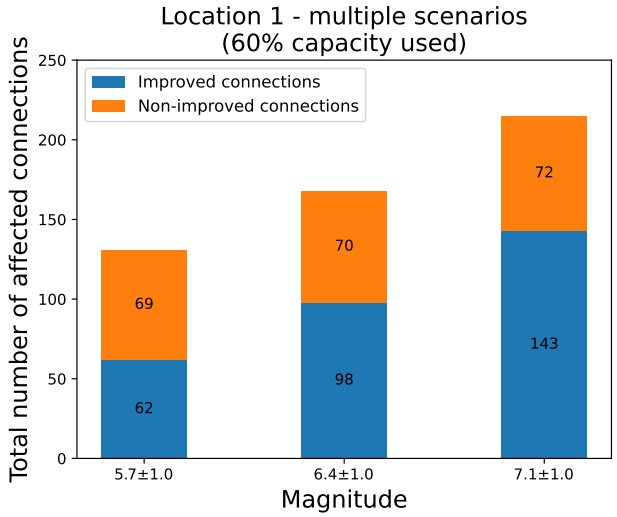
Figure 5.6 shows the total time needed to both compute and install the new paths for location 1 for all three provisioning strategies and unlimited capacity. As expected, for higher network utilization, the time increases. However, the difference between the utilization levels, including unlimited, is not remarkable. The runtime is mainly related to computation time, which is correlated



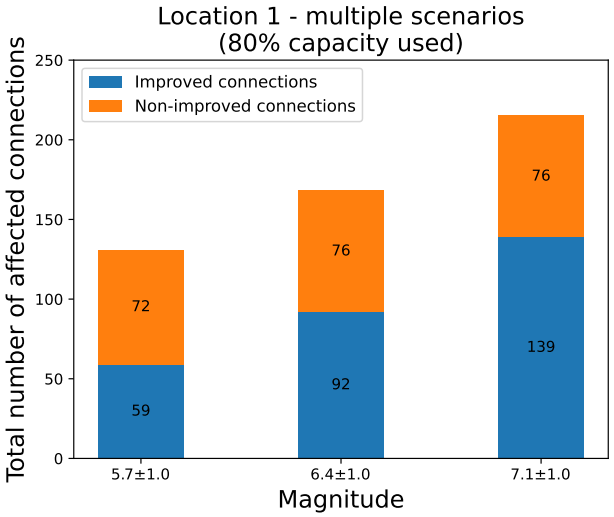
(a) Number of improved and non-improved connections at unlimited capacity utilization



(b) Number of improved and non-improved connections at 40% capacity utilization



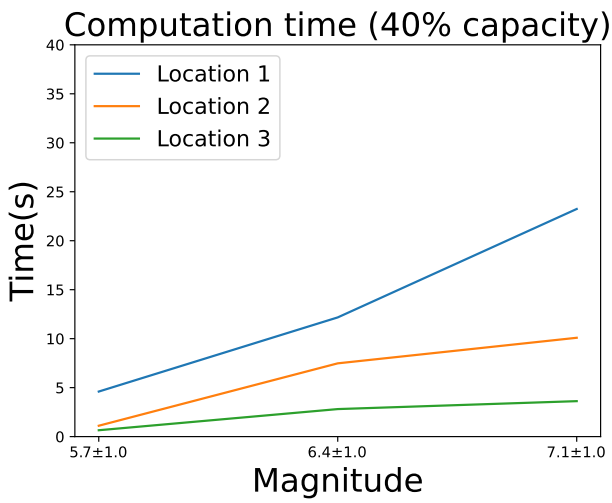
(c) Number of improved and non-improved connections at 60% capacity utilization



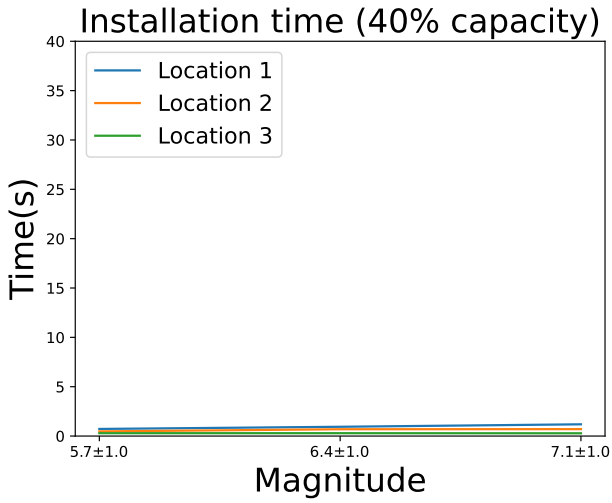
(d) Number of improved and non-improved connections at 80% capacity utilization

Figure 5.3: **Total number of affected connections and number of improved and non-improved connections for different capacity utilization.**

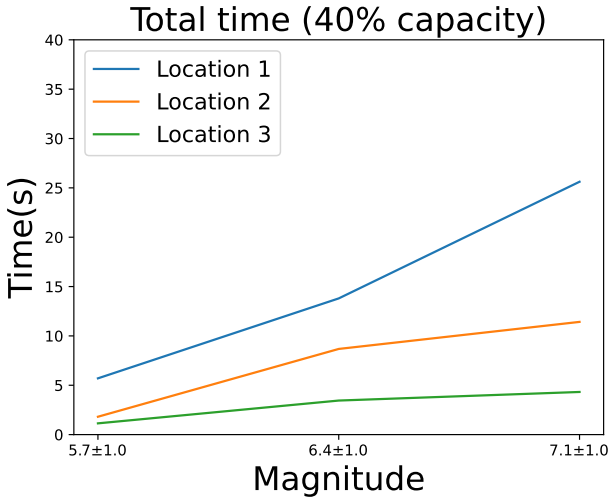
to the number of affected components and number of affected connections, respectively. Other factors, like provisioning strategy and installation time, are insignificant, and thus we can conclude that the utilization level is of minimal importance in terms of runtime. That is, however, influenced by our emulation environment, and installation times are expected to be more significant in actual



(a) Computation time



(b) Installation time



(c) Total time

Figure 5.4: **Time performance at 40% capacity utilization for all locations.**

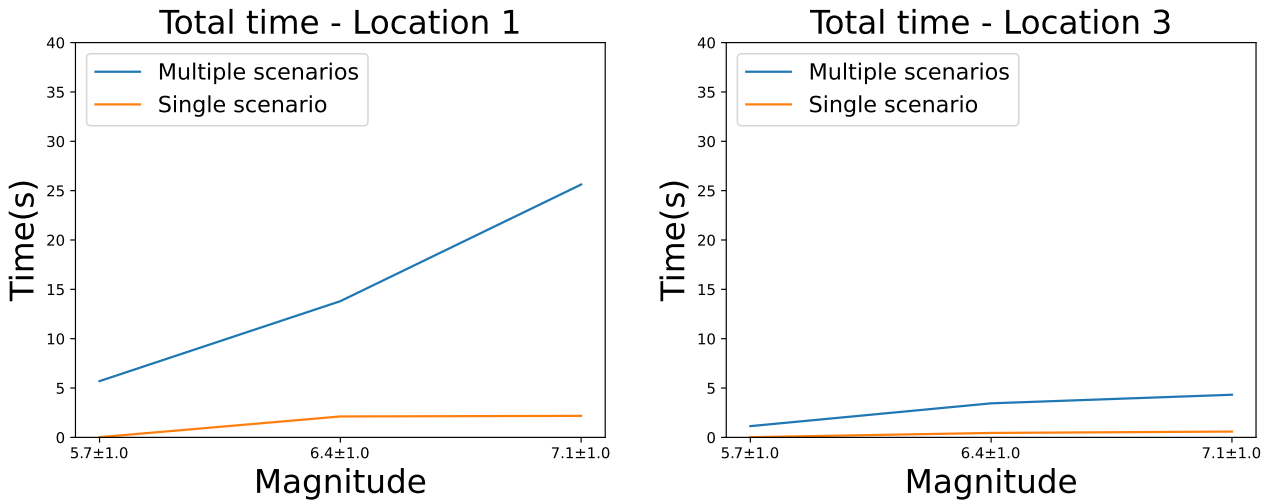
implementations.

5.2.4 Availability Improvement

To measure the improvement in the availability of the connection after finding an alternative path using the proposed algorithm, we take the average availability of the connections before rerouting and compare it to their availability after rerouting. The results for all utilization levels are presented in Figure 5.7. As expected, the newly computed paths' availability is higher than the original

		Single Scenario		Multiple Scenarios	
		Endangered connections	Saved connections	Endangered	Saved connections
Loc 1	5.7	0	0	131	110
	6.4	131	100	168	119
	7.1	131	102	215	144
Loc 2	5.7	0	0	60	60
	6.4	52	52	137	92
	7.1	60	60	137	92
Loc 3	5.7	0	0	30	30
	6.4	30	30	55	30
	7.1	35	30	55	30

Table 5.2: **The number of affected connections and the number of saved connections for all three locations, all three magnitudes, and for single and multiple scenario configurations.**



(a) Total runtime for multiple and single scenarios - Location 1. (b) Total runtime for multiple and single scenarios - Location 3.

Figure 5.5: **Total runtime for two locations at 40% capacity utilization**

paths. The average increase is 9.42% in Figure 5.7(a), 6.63% in Figure 5.7(b), 4.89% in Figure 5.7(c), and 4.00% in Figure 5.7(d); moreover, we notice that the improvement gains are increased with increased magnitudes of the earthquakes. That results from an increased drop in availability of the original scenario, and thus, there is more to be improved by our algorithm.

Furthermore, we can establish that the availability improvement is more significant for lower utilized networks. That can be explained with the simple understanding that when a network is utilized more extensively, there is less space for connections to be rerouted. That is proven by the fact that the improvement gain for lower magnitudes is comparable for all utilization levels but drops for higher magnitudes where more connections have to be rerouted, and less space is available.

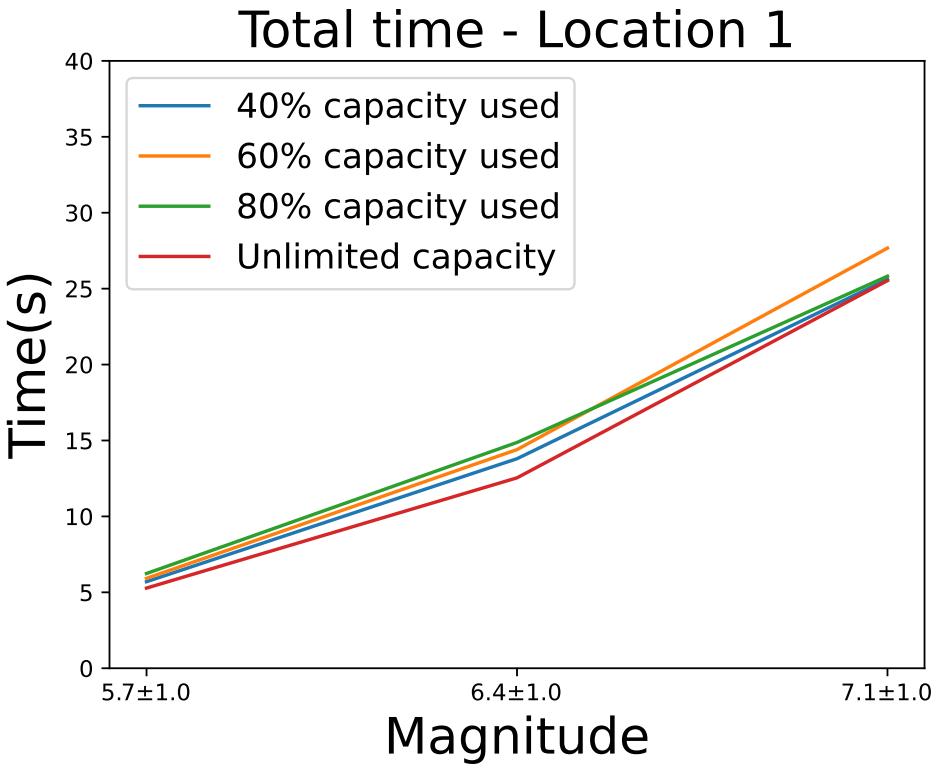
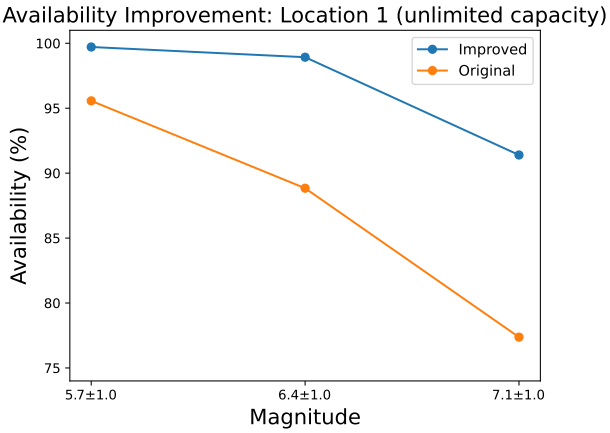


Figure 5.6: **Total time for Location 1 - different capacity utilization**

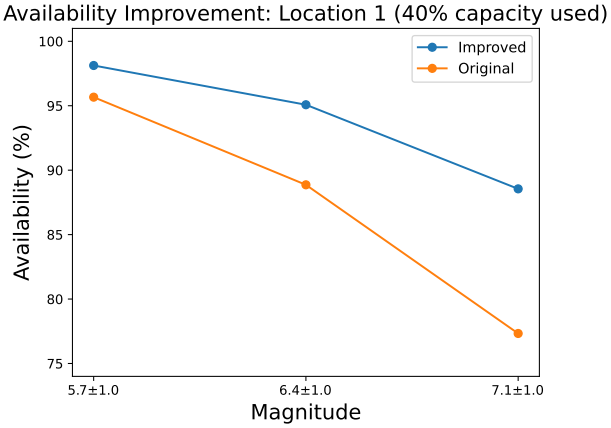
We can conclude that our rerouting algorithm is making significant improvements in availability and that this would be beneficial in an earthquake disaster. Especially in severe scenarios, our algorithm is making notable improvements.

5.2.5 Comparison between multiple and single scenarios

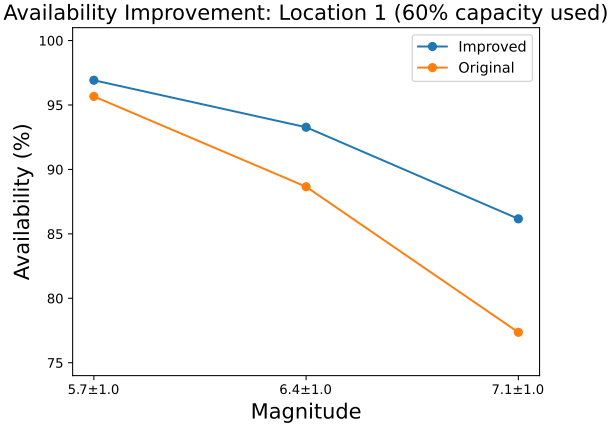
To verify the assumption that using multiple scenarios is beneficial compared to using a single scenario, we compare the availability of the newly computed paths. In Figure 5.8 we see the availability of the paths computed with a single scenario and multiple scenarios for different magnitudes across different utilization strategies at location 1. We see that the availability with multiple scenarios is higher than with a single scenario which validates our approach. Also, it is notable that our multi-scenario solution seems to outperform the single scenario more when the situation worsens. When the magnitude increases, the difference between the multiple and single scenarios gets larger. Finally, we also notice that when the network utilization increases, the solution with multiple scenarios has a lower drop in availability than the single scenario. We conclude that, even though the improvements are not always significant, the extra compute that it takes is worth the increase in availability.



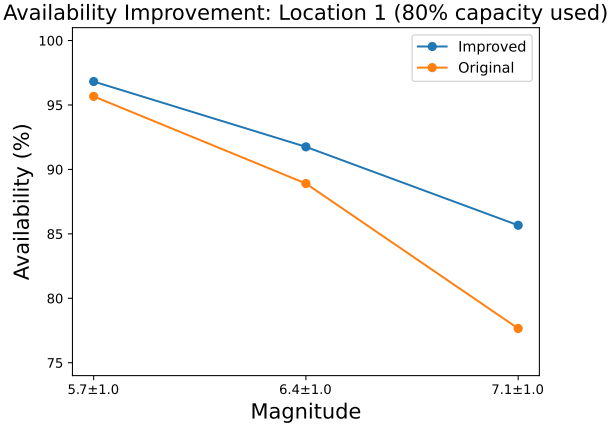
(a) Availability improvement at unlimited capacity



(b) Availability improvement at 40% capacity utilization



(c) Availability improvement at 60% capacity utilization

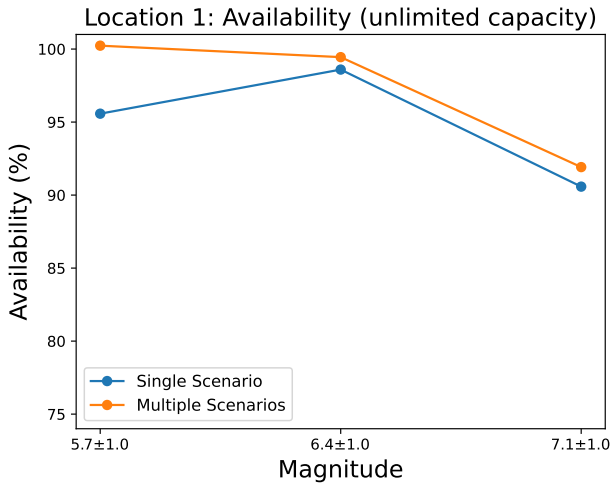


(d) Availability improvement at 80% capacity utilization

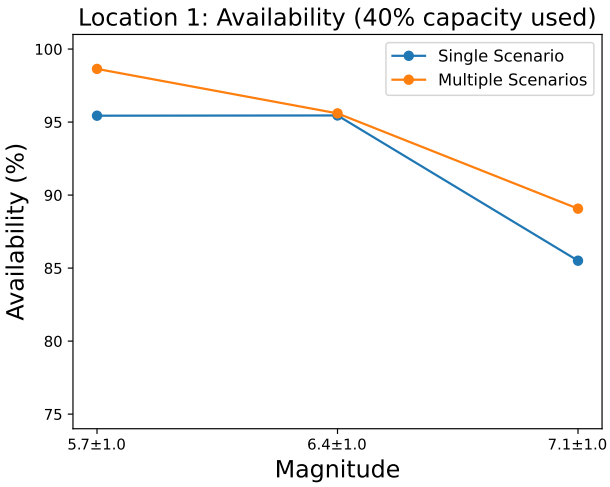
Figure 5.7: **Comparison of the availability of the new path vs. the availability of the original path at location 1.**

5.2.6 Bandwidth Retention

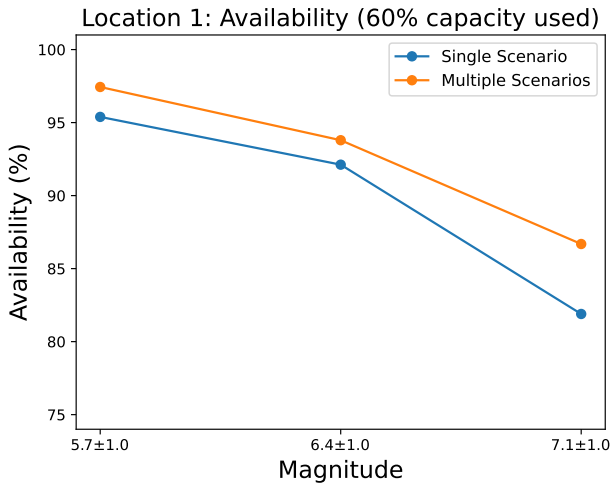
Figure 5.9 shows a comparison of the bandwidth retention in the network between our algorithm and a basic rerouting algorithm that does not include our bottleneck resolving algorithm. In these graphs, we show the percentage of bandwidth remaining after rerouting. To compute this bandwidth, we only count connections with a failure probability lower than our rerouting threshold. Figure 5.9(a) shows that the bandwidth that can be saved is strongly dependent on the free capacity that is available in the network. The common reasoning explains that more bandwidth can be retained when more space is available. More interesting is, however, Figure 5.9(b), here we see that if more traffic needs rerouting, the difference between our solution and a basic rerouting solution increases. We can conclude that our solution significantly increases the retained bandwidth in the network, especially for more severe disasters.



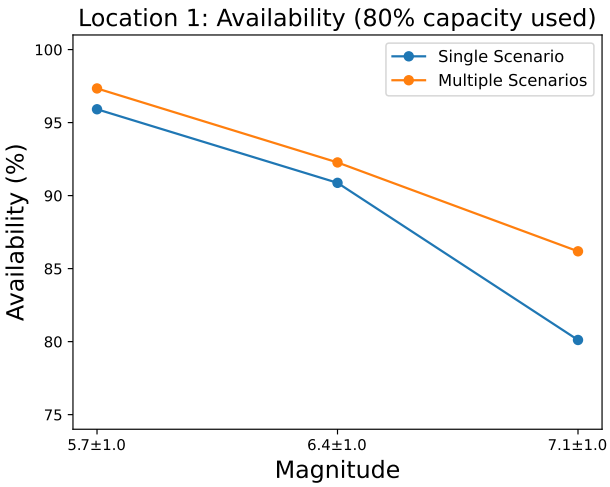
(a) Availability of the new path at unlimited capacity utilization



(b) Availability of the new path at 40% capacity utilization



(c) Availability of the new path at 60% capacity utilization

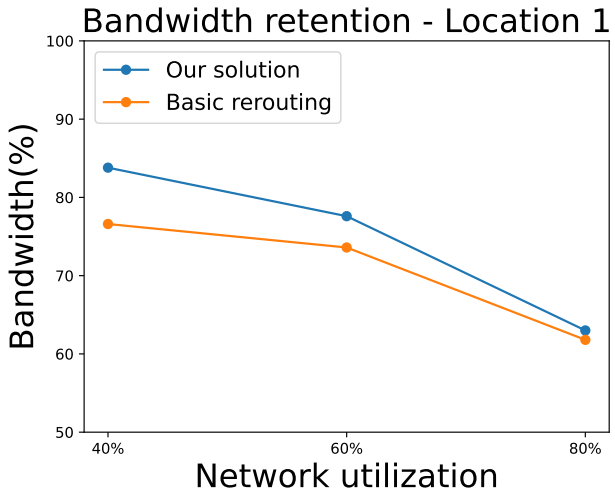


(d) Availability of the new path at 80% capacity utilization

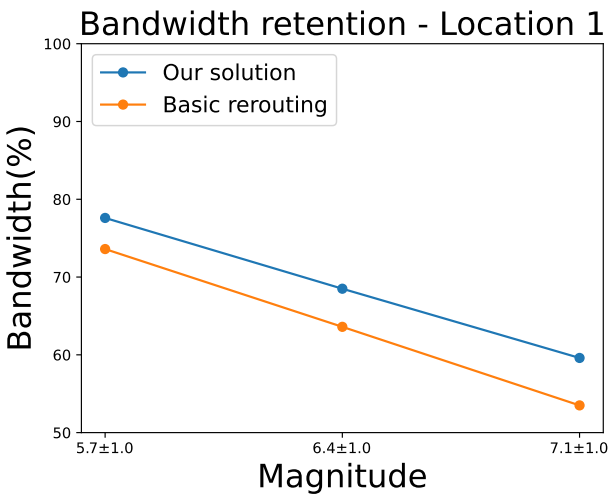
Figure 5.8: **Comparison between the availability of the alternative path - multiple vs. single scenario.**

5.3 Comparison between our solution and the exact solution

We expect our heuristic approach to provide a suboptimal solution to finding an alternative path. To check our approximation's accuracy, we will compare it to an exact solution. To find the exact solution, we implement a function to find all possible paths between a source and a destination node and evaluate their availability. We choose the path with the highest availability and compare it to the path we find using our heuristic. We expect that there will be a path



(a) Bandwidth retention at location 1 at 5.7 ± 1 magnitude



(b) Bandwidth retention at location 1 at 60% capacity utilization

Figure 5.9: **Comparison between the bandwidth retention of our algorithm vs. a basic rerouting algorithm.**

	Loc 1	**Loc2**	**Loc3**
5.7±1.0	0	0	0
6.4±1.0	0	45	0
7.1±1.0	16	45	0

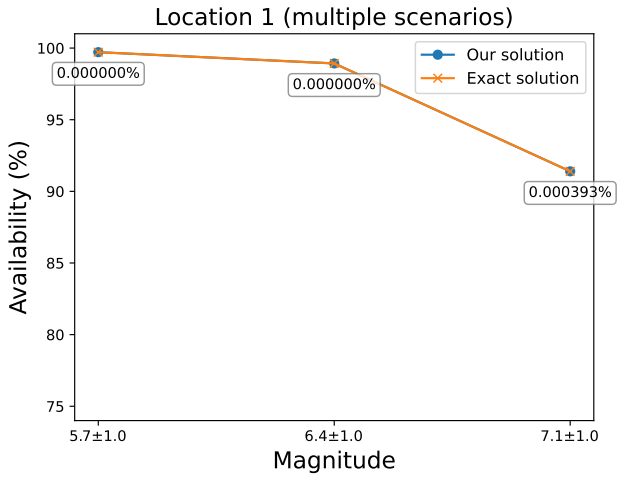
Table 5.3: **Number of connections with a higher-availability path.**

with better availability in some cases than the one we achieved; however, we expect this to come at the cost of very long computation times. We conduct experiments using three metrics to validate these expectations- path availability, computation time, and the number of connections for which a better path exists.

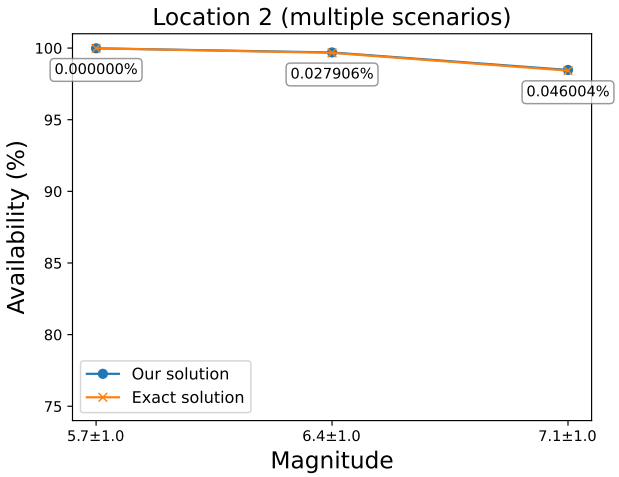
In Table 5.3, we see the number of connections for which using the exact solution provides a path with higher availability for all three test locations. Notably, for the lowest magnitude case, a better path does not exist for any location. At location 3, that is the case for all magnitudes.

5.3.1 Availability comparison

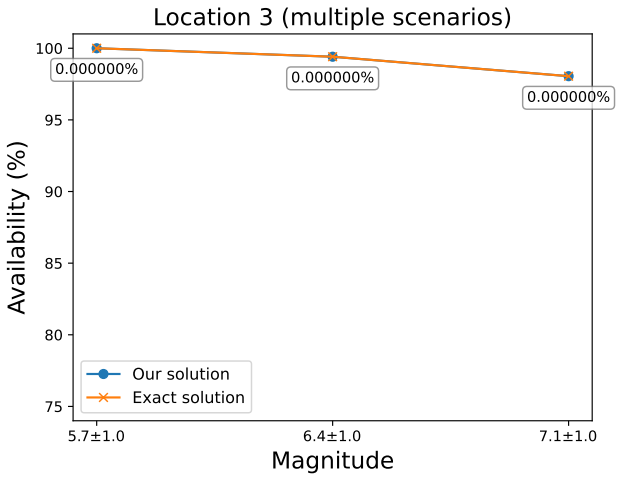
When we measure the availability of the alternative path computed with the proposed heuristic and the exact solution, we see that even though, in some cases, the exact solution provides a higher-availability path, the difference is minor (shown in Figure 5.10). From Table 5.3, we know that for location 2, we have the highest number of better paths. However, on average, the difference is about 0.02%, which is negligible, meaning that our solution is an excellent approximation and the computed paths have almost optimal availability.



(a) Location 1



(b) Location2

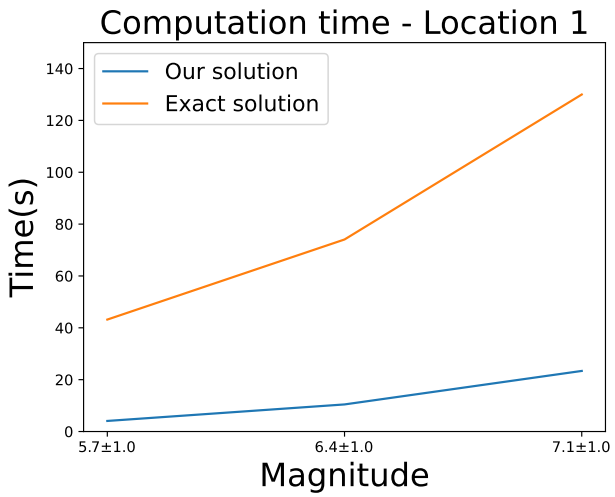


(c) Location 3

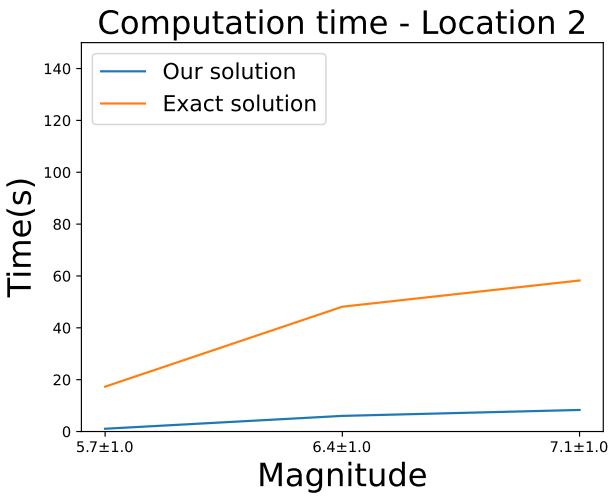
Figure 5.10: **Comparison between the availability of the new path using our solution against the exact solution for all locations and unlimited capacity.**

5.3.2 Computation time

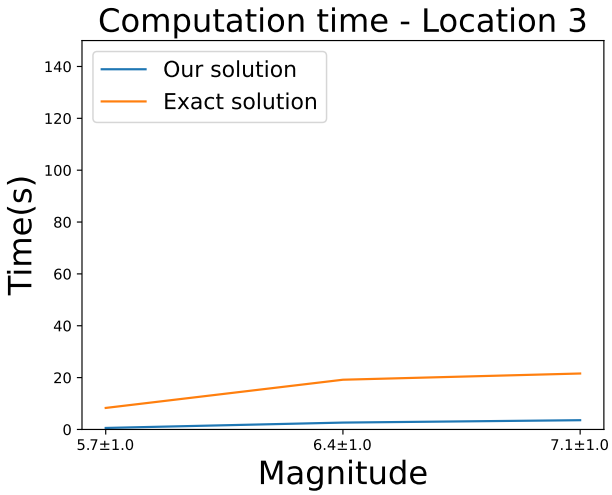
When we measure the time needed to find the exact solution, it is no surprise that it is significantly higher than our solution's runtime (see Figure 5.11). At location 1, finding the exact solution takes more than 2 minutes. Considering the results shown in Figure 5.10, we can conclude that our solution is very close to optimal at a much lower time cost.



(a) Location 1 unlimited capacity



(b) Location2 - unlimited capacity



(c) Location 3 unlimited capacity

Figure 5.11: **Comparison between the computation time using our solution against exact solution for all locations and unlimited capacity.**

5.4 Custom bandwidth & availability metric

As described in our second problem definition, we will judge our algorithm with our custom metric $A_p + B$, where B is the percentage of the bandwidth that is retained. We compare our approach with the basic algorithm used in Subsection 5.2.6, and the result of this is shown in Figure 5.12. We see that our solution scores better on our custom metric than the basic approach, proving that our solution is a step towards improving network reliability without the cost of an exact solution.

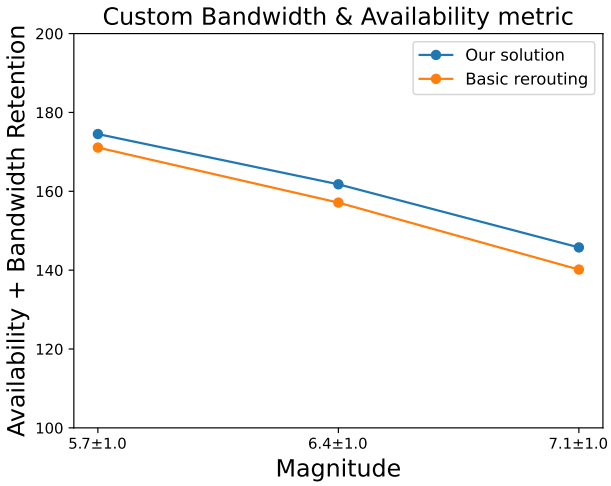


Figure 5.12: **Custom metric at location 1 at 60% capacity utilization.**

Chapter 6

Conclusion

Natural disasters are a known threat that disrupts network connectivity. Network protection strategies against disasters have been widely researched; however, it remains a challenge to create a practical approach. The issue comes from that that disasters like earthquakes are unpredictable and progress very rapidly after they occur. Even though early warning systems issue notifications when an earthquake starts, the available response time is strictly limited. Moreover, the information provided by the EWS is not entirely accurate. Under these conditions, applying a protection mechanism against a specific disaster model might not provide the desired result because the actual disaster might unfold unexpectedly.

We have proposed a solution that encompasses multiple scenarios generated based on the received early warning message to address the above issues. We use earthquake data to estimate the disaster area and assign failure probabilities to the components located in this area. Our work aims to save as many endangered connections as possible by routing them through alternative paths with maximized availability. In addition, we added a bandwidth constraint to get a more realistic scenario. That transformed our problem to finding a path with maximized availability, which does not exceed the available link capacity.

We proposed an algorithm for the defined problem that can identify bottleneck connections in a network and reroute these bottlenecks to create space for higher bandwidth connections. We assessed their performance by choosing three independent epicenter locations with different characteristics and performed the same experiments in all locations. The results showed that the run-time of our approach is fast enough to be performed within the bounds of the response window. We showed that our approach improves the availability of the connections by $\approx 9\%$ on average when the link capacity is unlimited. In addition, we demonstrated that having multiple scenarios provides higher availability, hence better protection than the single scenario case. We compared our heuristic to an exact solution and found that we could produce an excellent approximation in significantly less time. Finally, we combined connection availability and connection bandwidth into a custom metric that shows our algorithm is able to combine optimization for availability and throughput. To summarize, our work showed that the proposed solutions improve network availability within a reasonable time frame.

This work can now be used as a tool for network operators to minimize the

damage to critical infrastructure during a natural disaster, minimizing the devastation a natural disaster brings.

Chapter 7

Future Work

We created a network failure model with multiple mutually exclusive disaster scenarios and developed a protection strategy against them in the current work. We limited our scope to earthquakes; however, we can broaden the spectrum by adapting the system to other types of disasters such as hurricanes and tsunamis. That will require a different disaster model and a different type of early warning message.

Another interesting addition is to increase the number of disaster scenarios and analyze the trade-off between performance improvement and execution time. By adding more scenarios, the level of uncertainty will decrease; hence we can expect a better level of protection. However, this will eventually result in higher response times. Nonetheless, that might be less significant if we look at different disasters with longer warning times.

In our solution, we defined two problems- the second problem builds up on the first one by adding a bandwidth constraint, making our approach more realistic. We can expand this further by adding more constraints, valid in a real-world implementation. For instance, it will be worth considering the links' propagation delay and extending the problem to finding a least-delay alternative path.

Our experiments show that the time spent on installing flows, which also includes modifying flows, is not significant; however, we are performing the testing in an emulated environment. Therefore, the obtained results might differ when using actual hardware. Authors of [14] provide insights about the latency associated with adding, modifying, and deleting a flow entry. They consider factors such as the priority of the flow and flow table occupancy. From a time-efficiency perspective, it is reasonable to consider mechanisms for reducing the flow installation time. A way to achieve this is to adjust the algorithm such that first, we perform all path computations and then install/modify the flows. The advantage of this will be that only the final flow will be installed if a connection has to be rerouted multiple times. In addition to that, when we have multiple connections with equal bandwidth demand, and we have to choose one of them to be moved, we can select a connection that has been selected for rerouting in the past.

In terms of implementation, we proposed an SDN solution, and we assumed only one SDN controller is present, and it is outside of the disaster zone. We focused on protecting the data plane and not the control plane. Therefore, this

aspect can be further developed by looking into the deployment of redundant controllers to avoid a single point of failure. Furthermore, the location of the controllers can be chosen in a way that provides disaster resiliency [35].

Bibliography

[1] Richard M Allen. Probabilistic warning times for earthquake ground shaking in the san francisco bay area. *Seismological Research Letters*, 77(3):371–376, 2006.

[2] Richard M Allen and Diego Melgar. Earthquake early warning: Advances, scientific challenges, and societal needs. *Annual Review of Earth and Planetary Sciences*, 47:361–388, 2019.

[3] Reid Basher. Global early warning systems for natural hazards: systematic and people-centred. *Philosophical transactions of the royal society a: mathematical, physical and engineering sciences*, 364(1845):2167–2182, 2006.

[4] Hale Cetinay, Carmen Mas-Machuca, Jose L Marzo, Robert Kooij, and Piet Van Mieghem. Comparing destructive strategies for attacking networks. In *Guide to Disaster-Resilient Communication Networks*, pages 117–140. Springer, 2020.

[5] Andrew Coburn and Robin Spence. *Earthquake protection*. John Wiley & Sons, 2003.

[6] Geopy contributors. Geopy 2.2.0. `https://pypi.org/project/geopy/`.

[7] Mininet Project Contributors. Mininet. `http://mininet.org//`.

[8] Oscar Diaz, Feng Xu, Nasro Min-Allah, Mahmoud Khodeir, Min Peng, Samee Khan, and Nasir Ghani. Network survivability for multiple probabilistic failures. *IEEE Communications Letters*, 16(8):1320–1323, 2012.

[9] Ferhat Dikbiyik, Massimo Tornatore, and Biswanath Mukherjee. Minimizing the risk from disaster failures in optical backbone networks. *Journal of Lightwave Technology*, 32(18):3175–3183, 2014.

[10] Sifat Ferdousi, Massimo Tornatore, M Farhan Habib, and Biswanath Mukherjee. Rapid data evacuation for large-scale disasters in optical cloud networks. *Journal of Optical Communications and Networking*, 7(12):B163–B172, 2015.

[11] The Open Networking Foundation. Openflow switch specification version 1.3.0 (wire protocol 0x04). 2012.

[12] Teresa Gomes, János Tapolcai, Christian Esposito, David Hutchison, Fernando Kuipers, Jacek Rak, Amaro De Sousa, Athanasios Iossifides, Rui

Travanca, Joao André, et al. A survey of strategies for communication networks to protect against large-scale natural disasters. In *2016 8th international workshop on resilient networks design and modeling (RNDM)*, pages 11–22. IEEE, 2016.

[13] M Farhan Habib, Massimo Tornatore, Ferhat Dikbiyik, and Biswanath Mukherjee. Disaster survivability in optical communication networks. *Computer Communications*, 36(6):630–644, 2013.

[14] Keqiang He, Junaid Khalid, Aaron Gember-Jacobson, Sourav Das, Chaithan Prakash, Aditya Akella, Li Erran Li, and Marina Thottan. Measuring control plane latency in sdn-enabled switches. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, pages 1–6, 2015.

[15] Hirotada Honda and Hiroshi Saito. Nation-wide disaster avoidance control against heavy rain. *IEEE/ACM Transactions on Networking*, 27(3):1084–1097, 2019.

[16] Farabi Iqbal and Fernando Kuipers. Spatiotemporal risk-averse routing. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 395–400. IEEE, 2016.

[17] Alireza Izaddoost and Shahram Shah Heydari. A Probabilistic Model for Network Survivability in Large Scale Failure Scenarios. 2012.

[18] Alireza Izaddoost and Shahram Shah Heydari. Preventive network protection in probabilistic large-scale failure scenarios. In *2012 IEEE Globecom Workshops*, pages 858–862. IEEE, 2012.

[19] Alireza Izaddoost and Shahram Shah Heydari. Proactive risk mitigation for communication network resilience in disaster scenarios. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–4. IEEE, 2014.

[20] Alireza Izaddoost and Shahram Shah Heydari. Topology-based proactive network protection in large-scale failure scenarios. *Procedia Computer Science*, 34:111–117, 2014.

[21] Alireza Izaddoost and Shahram Shah Heydari. Risk-adaptive strategic network protection in disaster scenarios. *Journal of Communications and Networks*, 19(5):509–520, 2017.

[22] Moritz Kiese, Velislava Marcheva, Jörg Eberspächer, and Dominic Schupke. Diverse routing based on shared risk link groups. In *2009 7th International Workshop on Design of Reliable Communication Networks*, pages 153–159. IEEE, 2009.

[23] Hyang-Won Lee, Eytan Modiano, and Kayi Lee. Diverse routing in networks with probabilistic failures. *IEEE/ACM Transactions on networking*, 18(6):1895–1907, 2010.

[24] Lisheng Ma, Wei Su, Bin Wu, Tarik Taleb, Xiaohong Jiang, and Norio Shiratori. E-time early warning data backup in disaster-aware optical interconnected data center networks. *IEEE/OSA Journal of Optical Communications and Networking*, 9(6):536–545, 2017.

[25] Zenon Medina-Cetina and Farrokh Nadim. Stochastic design of an early warning system. *Georisk*, 2(4):223–236, 2008.

[26] Hiromitsu Nakamura, Shigeki Horiuchi, Changjiang Wu, Shunroku Yamamoto, and Paul A Rydelek. Evaluation of the real-time earthquake information system in japan. *Geophysical Research Letters*, 36(5), 2009.

[27] 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 & tutorials*, 16(3):1617–1634, 2014.

[28] Melike Oguz, Ferhat Dikbiyik, and H Serdar Kuyuk. Earthquake preparedness strategies for telecom backbone with integration of early warning systems and optical wdm networks. In *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 181–188. IEEE, 2016.

[29] Alija Pašić, Rita Girão-Silva, Ferenc Mogyorósi, Balázs Vass, Teresa Gomes, Péter Babarczi, Péter Revisnyei, Janos Tapolcai, and Jacek Rak. efradir: An enhanced framework for disaster resilience. *IEEE Access*, 9:13125–13148, 2021.

[30] Alija Pašić, Rita Girão-Silva, Balázs Vass, Teresa Gomes, and Péter Babarczi. Fradir: A novel framework for disaster resilience. In *2018 10th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 1–7. IEEE, 2018.

[31] Alija Pašić, Rita Girão-Silva, Balázs Vass, Teresa Gomes, Ferenc Mogyorósi, Péter Babarczi, and János Tapolcai. Fradir-ii: An improved framework for disaster resilience. In *2019 11th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 1–7. IEEE, 2019.

[32] Hiroshi Saito, Hirotada Honda, and Ryoichi Kawahara. Disaster avoidance control against heavy rainfall. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[33] S Sedef Savas, Ferhat Dikbiyik, M Farhan Habib, Massimo Tornatore, and Biswanath Mukherjee. Disaster-aware service provisioning with manycasting in cloud networks. *Photonic Network Communications*, 28(2):123–134, 2014.

[34] S Sedef Savas, M Farhan Habib, Massimo Tornatore, Ferhat Dikbiyik, and Biswanath Mukherjee. Network adaptability to disaster disruptions by exploiting degraded-service tolerance. *IEEE Communications Magazine*, 52(12):58–65, 2014.

[35] S Sedef Savas, Massimo Tornatore, M Farhan Habib, Pulak Chowdhury, and Biswanath Mukherjee. Disaster-resilient control plane design and mapping in software-defined networks. In *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6. IEEE, 2015.

[36] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43, 2013.

[37] J Tapolcai. regional-srlg. `https://github.com/jtapolcai/regional-srlg/tree/master/earthquake`, 2019.

[38] János Tapolcai, Balázs Vass, Zalán Heszberger, József Bíró, David Hay, Fernando A Kuipers, and Lajos Rónyai. A tractable stochastic model of correlated link failures caused by disasters. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2105–2113. IEEE, 2018.

[39] Phuong Nga Tran and Hiroshi Saito. Geographical route design of physical networks using earthquake risk information. *IEEE Communications Magazine*, 54(7):131–137, 2016.

[40] U.S. Geological Survey (USGS). Earthquake magnitude, energy release, and shaking intensity. `https://www.usgs.gov/programs/earthquake-hazards/earthquake-magnitude-energy-release-and-shaking-intensity`.

[41] U.S. Geological Survey (USGS). The modified mercalli intensity scale. `https://www.usgs.gov/natural-hazards/earthquake-hazards/science/modified-mercalli-intensity-scale/`.

[42] U.S. Geological Survey (USGS). Search earthquake catalog. `https://earthquake.usgs.gov/earthquakes/search/`.

[43] Alessandro Valentini, Balázs Vass, Jorik Oostenbrink, Levente Csák, Fernando Kuipers, Bruno Pace, David Hay, and János Tapolcai. Network resiliency against earthquakes. In *2019 11th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 1–7. IEEE, 2019.

[44] Xiaokang Xie, Qing Ling, Ping Lu, Wei Xu, and Zuqing Zhu. Evacuate before too late: Distributed backup in inter-dc networks with progressive disasters. *IEEE Transactions on Parallel and Distributed Systems*, 29(5):1058–1074, 2017.

[45] Song Yang, Stojan Trajanovski, and Fernando A Kuipers. Availability-based path selection and network vulnerability assessment. *Networks*, 66(4):306–319, 2015.