UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

FAST REROUTE USING SEGMENT ROUTING FOR SMART GRIDS

Documento Provisório

Frederico António Silva de Brito

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

Dissertação orientada por:
Prof. Doutor Fernando Manuel Valente Ramos
e co-orientado pelo Prof. Doutor Nuno Fuentecilla Maia Ferreira Neves

2016

Agradecimentos

Em primeiro lugar quero agradecer aos meus pais, por todos os esforços que fizeram para eu me encontrar aqui, por me passarem todos os seus valores, amor e carinho. Nada disto poderia ter acontecido sem vocês. Quero também agradecer aos meus restantes familiares por estarem presentes e por me apoiarem nesta jornada.

Em segundo lugar quero agradecer muito, mas mesmo muito aos meus amigos e colegas de curso: Filipe Custódio, João Rebelo, Joel Alcântara, José Soares e Luís Ferrolho. Muitas manhãs, muitas tardes e muitas noites de dores de cabeça e estudo mas também de muita conversa, gargalhadas e episódios históricos. Um especial obrigado para o meu amigo Rebelo, pela brilhante decisão de nos inscrevermos na FCUL e pela ajuda dada no primeiro ano. Não foi fácil, mas já está. Às pessoas que conheci na FCUL e se tornaram meus amigos, e alguns bem grandes, um muito obrigado.

Finalmente, aos meus orientadores, Prof. Fernando Ramos e Prof. Nuno Neves, por me terem proporcionado esta oportunidade, pela forma como trocamos ideias e pelo conhecimento passado. A eles o meu muito obrigado. Além disso queria também agradecer ao financiamento que foi suportado pelo projecto SEGRID e também pela FCT através do programa Multianual do LASIGE.

Aos demais que de alguma forma possam ter contribuído para o meu sucesso, o meu mais sincero obrigado.

A toda a minha família e amigos.

Resumo

A rede eléctrica tem contribuído de forma extraordinária para o nosso dia-a-dia nas últimas décadas e, como tal, tornou-se essencial para a nossa sociedade. Hoje em dia, estão a ser tomadas decisões para a modernizar, de modo a que seja possível fornecer novos serviços. Graças ao aumento da produção de electricidade através de energias renováveis (energia solar, hídrica e eólica), e ao aumento do consumo de energia, é vista como necessária uma reestruturação da rede eléctrica. Para atingir estes objectivos, foi proposta uma nova geração destas redes, as Smart Grids (SG). As SG são compostas por dispositivos electrónicos inteligentes, sensores com e sem fios e contadores inteligentes que necessitam de se coordenar para funcionarem correctamente. Como tal, é fundamental ter uma rede de comunicação moderna capaz de suportar estes desafios [1].

Um conjunto de propriedades de que se destacam a escalabilidade, disponibilidade e segurança, são fulcrais para o funcionamento das SG. Para as SG a infra-estrutura de comunicação tem um papel particularmente importante para que se possam cumprir estas necessidades. As tecnologias actuais baseadas em Internet Protocol (IP) e em Multiprotocol Layer Switching (MPLS) têm conseguido corresponder a estas necessidades. O protocolo IP é um dos alicerces para a comunicação mundial, enquanto que o MPLS tem sido adoptado pelas suas capacidades de engenharia de tráfego.

No entanto, as redes de IP tradicionais são difíceis de gerir e tornam complicado o desenho de soluções que permitam utilização eficiente de recursos e que possibilitem comunicação resiliente. Por outro lado, o MPLS tem problemas de escalabilidade devido ao uso de protocolos complexos como o Resource Reservation Protocol with Traffic Engineering (RSVP-TE).

As Software Defined Networks (SDN) promete resolver alguns dos problemas mencionados anteriormente, a partir do desacoplamento do plano de dados do plano de controlo, que passa a ser gerido por um controlador logicamente centralizado [2][3][4]. Deste modo, as aplicações que são executadas no controlador têm uma visão centralizada do estado da rede, o que facilita a procura de soluções de gestão de redes.

No entanto, os operadores de SG poderão apresentar alguma relutância ao mover todos os seus elementos da rede para uma SDN. Felizmente, foi proposto recentemente um novo protocolo pela Internet Engineering Task Force (IETF) – Segment Routing (SR) [5] – que permite a centralização lógica oferecida por uma SDN num ambiente de uma

rede MPLS. SR é muito semelhante ao MPLS, na medida em que utiliza segmentos que se comportam como etiquetas MPLS. A comutação de pacotes, baseada também nestas etiquetas, é gerida por comutadores que usam as mesmas acções do MPLS (push, pop e swap).

No entanto, ao contrário do MPLS, o SR não necessita de protocolos complexos como o RSVP-TE, simplificando a gestão da rede. O SR utiliza uma forma de source routing, facilitadora da sua integração. Desta forma o SR pode ser integrado com os controladores SDN e outras aplicações. Para implementar SR, o controlador SDN apenas precisa de enviar uma lista ordenada de segmentos para o encaminhador que a insere no cabeçalho dos pacotes quando necessitarem de serem enviados. Isto torna possível a criação de uma solução mais simples e escalável para engenharia de tráfego.

Nesta tese vamos explorar o uso de SR para avaliar a resiliencia da rede. O objectivo passa por desenhar e avaliar as soluções que forneçam reencaminhamento rápido após uma falha de uma ligação entre nós. Em particular, fornece a capacidade de realizar reencaminhamento rápido enquanto fornece uma grande percentagem de cobertura. Aproveitando as caracteriscas das SDN e de SR, as nossas soluções permitem que o controlador pré compute os caminhos de backup necessários para instalar nos encaminhadores, mantendo o plano de dados em MPLS inalterado.

A contribuição principal desta tese pode ser resumida em dois pontos:

1. Desenho de uma solução de reencaminhamento rápido em caso de faltas para Smart Grids, usando SR e SDN.

2. Fornecer uma avaliação exaustiva do algoritmo de modo a que se consiga compreender os seus benefícios e limitações.

O algoritmo proposto utiliza vários comutadores que são utilizados como destinos intermédios, que garantem a entrega dos pacotes após a falha de uma ligação entre nós. Como tal, também propomos dois selectores de segmentos que fornecem reencaminhamento rápido mas com características diferentes.

A primeira solução, Fast Segment Drop (FSD), selecciona um segmento próximo da origem do caminho em vez do segmento mais próximo do destino. Isto permite que os pacotes que atravessam a rede causem o menor overhead possível. O overhead deve-se ao número de segmentos usados em cada nó durante o caminho. Assim sendo, se escolhermos um segmento mais próximo do destino o overhead será maior.

A segunda solução, Congestion Avoidance Segment (CAS), escolhe segmentos que podem aumentar o overhead mas que, em contraste, fornecem a capacidade de escolher o caminho com menor utilização. Deste modo pode-se evitar estrangulamentos existentes na rede.

Para compararmos as nossas soluções implementamos um selector aleatório e o algoritmo TI-LFA [6]. Os resultados demonstram que para a maioria das topologias uma

falha entre nós pode ser tolerada utilizando Loop Free Alternatives (LFA). No entanto ainda existem cerca de 20% dos casos que necessitam de utilizar um segmento para tolerar uma falha, enquanto que dois segmentos raramente são necessários. Também foi possível concluir que o nosso algoritmo fornece mais flexibilidade na escolha de segmentos do que TI-LFA visto que permite uma maior escolha de segmentos. Utilizando CAS é possível reduzir ligeiramente a congestão das ligações na rede em grids e em topologias reais.

Palavras-chave: SDN, Segment Routing, engenharia de tráfego, encaminhamento resiliente, balanceamento de carga, Smart Grid.

Abstract

With the increase of power generation from renewable sources and with a growing energy demand, the traditional communication network underpinning the actual electric power grid needs an overhaul.

As a response, the Smart Grid is a new generation of electric grids that aims to fulfill this goal. Smart Grids demand a set of properties that range from high availability to scalability and security. Therefore, the communication infrastructure plays an important role. Current Internet Protocol-based and Multiprotocol Layer Switching (MPLS) technologies have been suggested capable in achieving those needs. However, IP networks have problems to offer traffic engineering solutions and MPLS faces scalability problems due to the use of complex protocols such as RSVP-TE.

A new network paradigm, Software-Defined Networks (SDN), is revolutionizing the way computer networks are built and operated, and is leading to the "softwarization" of networking. Showing promise to solve some of the above problems. However, smart grid operators may be reluctant to move all their network elements to SDN anytime soon. Fortunately, Segment routing, recently proposed by the IETF, allows SDN to be used in the context of MPLS networks. The data plane of Segment Routing is similar to MPLS as it uses segments that behave as MPLS labels and is managed in switches using similar actions.

In this thesis we present algorithms for fast reroute in SR networks. We propose two solutions: Fast Segment Drop (FSD) that aims to minimize packet overhead and segment list size; and Congestion Avoidance Segment (CAS), a solution that provides traffic engineering by minimizing the maximum link load. The results indeed show that by using CAS reduces network congestion when compared with other algorithms. FSD provides higher coverage using just one segment thus reducing overhead.

Keywords: SDN, Segment Routing, Traffic Engineering, Resilient Routing, Smart Grid, Load balancing

Contents

List of Figures

List of Tables

Chapter 1 - Introduction

The electrical power grid is central to our society, having contributed greatly to our daily life for the past decades. Nowadays, important steps are occurring towards the modernization of the grid to accommodate newer services. With the increase of power generation from renewable sources, like solar, wave and wind energy, and with a growing energy demand (e.g., electric cars), the traditional power grid needs an overhaul. To fulfill this goal, a new generation of electrical infrastructure, called the Smart Grid (SG), is being deployed, composed by a complex network of intelligent electronic devices (IED), wired and wireless sensors, smart meters and dispersed loads, that require coordination in order to work properly. Therefore, a modern communications network is fundamental to support the new challenges brought up by the SG [1].

The SGs demands a set of properties that include timeliness, scalability, high availability, and security. Besides, they also need to be interoperable, able to connect and exchange data freely and transparently with many different types of devices. The communication infrastructure plays an important role in fulfilling those needs. In particular, current Internet Protocol-based (IP) and Multiprotocol Layer Switching (MPLS) backbone technologies have been meeting these needs fairly well. IP is the interoperable foundation for communications around the world, while MPLS is being chosen to implement Traffic Engineering solutions [7][8].

However, traditional IP networks are hard to manage and have difficulties to offer solutions with efficient resource utilization for resilient communication. In addition, MPLS faces scalability problems due to the use of complex protocols like Resource Reservation Protocol with Traffic Engineering (RSVP-TE).

Software Defined Networking (SDN) shows promise to solve some of the above problems by decoupling the data plane and the control plane of networks, managed in this new paradigm by a logically centralized controller [2][3][4]. As such, it provides network applications running in the controller with a centralized view of the distributed network state, facilitating the construction of solutions for network management. However, smart grid operators may be reluctant to move all their network elements to SDN anytime soon.

Fortunately, a new network technology has been proposed recently by the Internet Engineering Task Force (IETF) – Segment Routing (SR) [5] – which allows SDN to be used in the context of MPLS networks. SR data plane is similar to MPLS as it uses segments that behave as MPLS labels and is managed in switches using similar actions (push, pop and swap). However, unlike MPLS, it does not require the use of complex protocols such as RSVP-TE, simplifying the management tasks.

SR thus enables traffic engineering and advanced routing services in IP networks while preserving compatibility with a traditional MPLS data plane. With this technique, the core MPLS data plane *does not have to be changed* and SDN can control the network from the edge. Taking advantage of the idea, we study and implement an approach that makes the network more robust to failures. In particular, it gains the ability to perform fast reroute and recovery after link failures.

1.1 Motivation

In the past decades, electrical grids have suffered from several blackouts that have disrupted their service. One of the reasons for most blackouts is the slow response times of devices over the grid. This is usually the result of problems with the communications used among the different components in current grids. Considering that the SG requires high levels of reliability in packet delivery, a simple disruption of a link in a part of the network has the potential to affect hundreds of thousands of connections. Several techniques to solve this problem have been proposed. The most effective is to install precomputed multiple redundant paths on switches, instead of recomputing the paths after a failure. This provides fast rerouting enabling rapid recovery after link failures.

According to [7][8], most smart grid communication requirements can be provided by IP and MPLS protocols. IP is the main communication protocol used in the network layer [9]. It offers best effort transmission of blocks of data called datagrams from sources to destinations. IP is widely used in our daily life and has proven to be scalable and efficient. However, IP solutions have limited support for traffic engineering and fault tolerance requires routers to perform complex tasks, such as tunneling and direct forwarding, which are typically not available in common routers or are too expensive for the network as a whole. Alternatively, MPLS is commonly used to implement traffic engineering [10]. In a MPLS network, each router independently chooses a next hop for a packet based on the analysis of the packet's label, a short fixed length value. For this purpose, MPLS requires a Label Distribution Protocol (LDP) that uses a set of procedures by which one router informs another of the label bindings it has made. In addition to LDP, MPLS uses RSVP-TE, another complex protocol, that allows for traffic engineering and fast reroute. Using these protocols increases the complexity of the network and limits its scalability. Namely, RSVP-TE inherites some properties of RSVP that adversely affect its control plane scalability, like reliance on periodic refreshes for state synchronization.

Two recent innovations in computer networking, SDN [2] [3] [4] and SR [5], show promise to solve the above problems. SDN decouples the data plane and the control plane, which is now managed by a logically centralized controller providing applications a centralized view of the network state. SR is a novel protocol, that improves traffic engineering and allows fast fault recovery for IP networks, while maintaining the same MPLS data plane. However, it does not require the use of protocols such as RSVP-TE or LDP, simplifying the control plane.

1.2 Software Defined Networks and Segment Routing

The layered structure of the internet protocol stack had a major contribution for its success. It allowed data plane services to be broken down into their fundamental components and consequently offered innovation in each layer. However, problems come from the complexity of the control plane of computer networks. The reason for this complexity is derived from the coupling of the control and data planes inside the network equipment. This makes the development and deployment of new networking features very hard because these normally would require a modification of all network devices.

With SDN it is possible to decouple the control and the data plane[2] [11]. The control functionality is moved to the SDN controller, leaving the network devices with the forwarding functions only. The controller is a platform, logically centralized, that runs on commodity servers. This provides the resources and abstractions to program the network devices (e.g., switches or routers). Forwarding decisions are flow-based, instead of destination-based and all packets of a flow receive identical policy decisions at the network devices.

SR improves traffic engineering in IP networks while preserving compatibility with traditional MPLS data plane [5]. Similar to MPLS the idea is to break up the routing path into labels (segments) in order to control routing paths. In SR there are two types of segments: Node-id segments that represent switches in the network, and adjacency segments that identify the links connecting a node and its neighbors. These segments are encoded as a MPLS label, which are inserted in the label stack of the packet header, also known as Segment List (SL) allowing a source routing approach. Each packet is then forwarded along the shortest path towards the network element represented by the top segment. When a packet reaches the intermediate destination represented at the top of the segment list, the top level label is popped by the intermediate destination and the packet is forwarded to the next segment in the list. This actions continues until the packet reaches its final receiver. As such, the stack of labels allows customized routing.

SR can be integrated with SDN. The SDN controller can learn the network topology and the real time state information. Using this information, the SDN controller can calculate the best network path based on a set of predefined criteria. To implement SR, the SDN controller needs only to send an ordered list of labels to the source router, which should be inserted in the header when the packet is locally forwarded. This provides a much more scalable and simple solution for traffic engineering. Millions of applications or flows can have millions of different network paths.

1.3 Goals & Contributions

Critical services of the Smart Grids that require high reliability are being deployed in IP networks. A single link failure can disrupt many connections causing congestion and loss of data, which may prevent for instance, the remote control of distant devices a serious problem in this infrastructure. A technique that may mitigate this problem is Fast Reroute. By installing precomputed redundant paths on switches, it allows them to recover quickly when a failure occurs since they no longer have to recompute paths after that failure. As such, recovery time are reduced from hundred of milliseconds or seconds.

The goal of our work is to design and evaluate a solution that provides fast reroute after a single link failure for Smart Grids. We leverage on an SDN controller to precompute the paths and on SR to steer traffic while leaving keep the MPLS data plane unchanged. We also aim to provide solutions that offer good trade-offs between packet overhead and load balancing.

The main contributions can be summarized as:

1. Design of a fast reroute solution for Smart Grids, using SR and SDN.

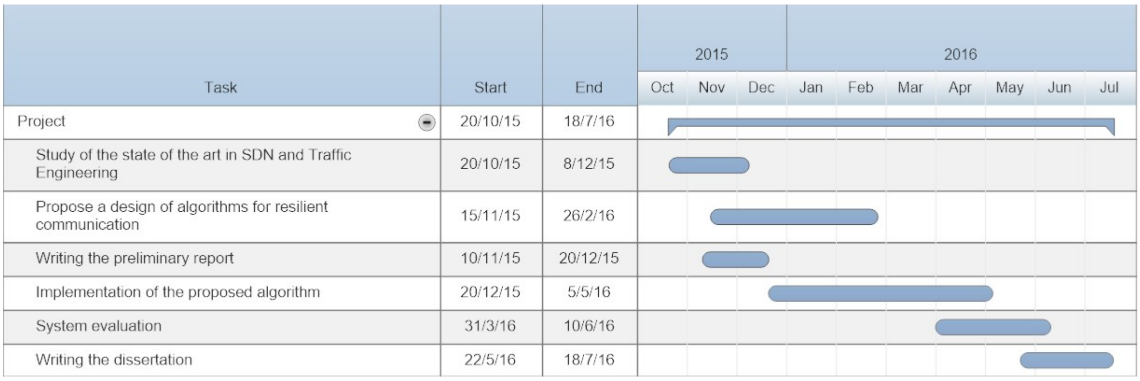
2. Provide an extensive evaluation of the algorithm to better understand its benefits and limitations.

The proposed algorithm may employ different strategies to ensure delivery after a single link failure. We propose two strategies that provide fast reroute but with different properties.

The first solution tries to minimize the number of segments used. For this purpose, it prefers to forward packets to switches closer to the source when failure occurs since, this allows packets to traverse the network while causing less overhead. The overhead is the cumulative number of segments used at each hop along the path.

The second solution opts to balance load by choosing the path with minimum link load, possibly avoiding choke points in the network.

1.4 Work Planning



Task	Start	End
Project	20/10/15	18/7/16
Study of the state of the art in SDN and Traffic Engineering	20/10/15	8/12/15
Propose a design of algorithms for resilient communication	15/11/15	26/2/16
Writing the preliminary report	10/11/15	20/12/15
Implementation of the proposed algorithm	20/12/15	5/5/16
System evaluation	31/3/16	10/6/16
Writing the dissertation	22/5/16	18/7/16

Figure 1.1: Working Plan.

This work was part of the SEGRID (Security for Smart Electricity GRIDs) project, funded by the EU under the FP7 program. FCUL is one of its partners along with distribution system operators, manufacturers, research institutions and universities. The project main objective is to increase smart grid security from cyber-attacks. FCUL is responsible for providing a resilient communication infrastructure and this thesis is part of that effort. The work described in this thesis contributed to the deliverable do SEGRID - Deliverable 4.2, Preliminary specification of security and privacy solutions.

We elaborated a plan to achieve the established goals in the beginning of the project. The plan was divided in the following tasks:

1. Study of the state of art on SDN, Traffic Engineering and Routing – with the goal of investigate different techniques that could be explored to make our algorithm for resilient communication.

2. Design of algorithms for resilient communication – based on the techniques investigated while studying the state of the art, and novel techniques required for our project.

3. Writing the preliminary report.

4. Implementation of the proposed algorithms.

5. System Evaluation – including analysis of performance and comparison with other algorithms.

6. Writing the dissertation.

Most of planned tasks were accomplished in their due time. However, the implementation task took longer then expected due to learning new tools. There was also a small delay in designing the load balance solution.

1.5 Document Structure

The rest of the document is organized as follows. Chapter 2 addresses the background and related work. In this chapter we present an introduction to Smart Grids, SDN, and study of communication networks with a focus on resilience. In Chapter 3 we present our proposal for fast reroute. Chapter 4 covers the evaluation of our solution. Finally, Chapter 5 summarizes and concludes this work.

Chapter 2 - Background and Related Work

The expansion of the electric grid with new energy generation resources and customers demanding more electricity, created the need for the deployment of the SG. These infrastructures are composed of several devices such as smart meters, data concentrators and various kinds of servers that need to be reliably connected through a network.

This section provides an overview of the communication architecture within a SG, the types of wired and wireless technologies that are employed, and also the most suited communication protocols. Some background information on communication networks is also included. We start by providing basic principles of routing and forwarding. There are several protocols that allow gateways to exchange routing information, such as Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS). Despite its wide utilization, OSPF is not flexible enough to support traffic engineering for diverse applications. Some enhancements have been proposed to enable OSPF to provide traffic engineering by changing link weight values, but even these approaches may cause congestion and unbalance traffic. MPLS was proposed to address these issues. However, MPLS traffic engineering may suffer from scalability and robustness problems due to the use of tunnels, and cannot offer service differentiation.

SDN shows promise to solve the above limitations by decoupling the data plane and the control plane. The section on SDN provides information related to its architecture and protocols. In addition, it presents SR, its advantages and explains how it can be integrated in a SDN network.

Finally, the last section describes several resilient routing techniques that allow the traffic to flow even in the event of a failure, including IP and MPLS resilient routing techniques through fast reroute mechanisms.

2.1 Smart Grids

SG is a new generation of electric grids, that are composed of IEDs, smart meters, wired and wireless sensors, and dispersed loads that require cooperation and coordination in order to work properly [1] [12] [13] . They offer many benefits to utilities and consumers – mostly seen in energy efficiency improvements on the electricity grid and in the energy users' homes and offices.

There is not a single solution or a representative network for the SG as each utility may require different topologies, policies and application requirements. However, SG are normally multi-tier networks supported by several different communication technologies to enable efficient and reliable access to grid components.

In general, the SG communication architecture can be divided in different areas, such as depicted in Figure 2.1: Home Area Networks (HAN); Neighborhood Area Networks (NAN); Field Area Networks (FAN); and Wide Area Networks (WAN).

2.1.1 Home Area Networks

A HAN is a communication network of appliances and devices within a home. They are used to monitor and control how devices implement new functionalities like the Advanced Metering Infrastructure (AMI) and Demand Response (DR). The idea is to have every home device send its power readings over this network to the smart meter, which will eventually send the readings to the control center. The data generated from each in-home appliance and the communications requirements of each device in a HAN may differ. For instance, a laptop computer does not cause a significant load on the electrical network and therefore control centers only need simple information. The data rate expected in a HAN is low, typically around 1–10 Kbps.

A HAN can also be a Business Area Network (BAN) if it is used to refer to networks implemented in a company, and an Industrial Area Network (IAN) when applied to an industrial setting.

2.1.2 Neighborhood Area Networks and Field Area Networks

A NAN connects smart meters to a data concentrator of the AMI. It can form a wireless mesh network due to the large number of devices (e.g., smart meters and data collectors) spread across a respective zone. A FAN forms the communication facility inside an electrical station, such as a wind power plant.

NANs and FANs have the same objective, to share and exchange information, of particular types of applications: field based (e.g., to transmission lines, sensors, voltage regulators) and customer based (e.g., to end customers, like houses, buildings, industrial users). Customer based applications (i.e., AMI, DR) require the communication network between the customers and the utility to be highly scalable, allowing for further addition of applications and customers. On the other hand, field based applications (i.e., SCADA) are normally more time sensitive due to their need to react to events within well defined intervals. Usually the data rate in NANs is higher than that of HANs.

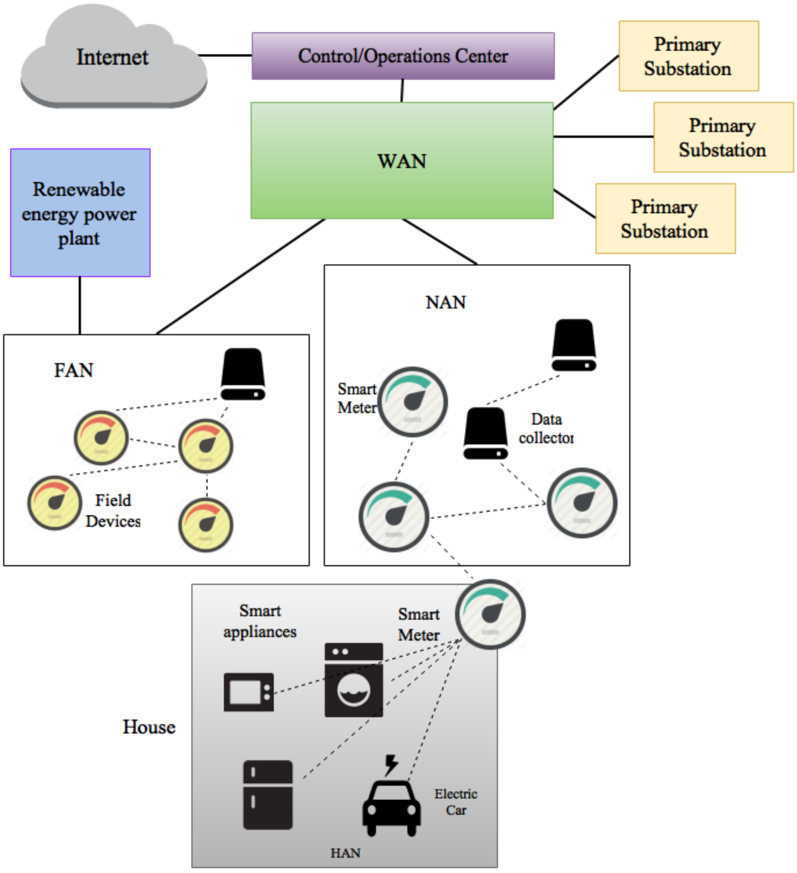


Figure 2.1: Multi-tier smart grid communications network with HANs, NANs, FANs and WANs.

2.1.3 Wide Area Networks

Wide area networks form the communication backbone that connects the highly distributed smaller area networks that serve the power systems at different locations. WAN comprises two types of networks: Core and Backhaul. While the core network is used to connect the primary substations which are integral parts of a power system and form important links between the generation stations, the backhaul network is used to connect the NAN to the core network. The real-time measurements taken by IED's and electric devices are transported to the control centers through the WAN, which usually requires high bandwidth to support all the communications needed from all the devices.

2.1.4 Wired and Wireless technologies

SG networks are potentially spread over wide geographic areas. Therefore, a range of wired and wireless technologies are used. However, none of them suits all the applications, and thus it is necessary to fit the best technology for a respective group of power system applications.

The Power Line Communication for instance, is a wired technology based on electricity wires. It can be used to carry information, but the electrical wire circuits have limited ability to carry higher frequencies. In practice, Power line communication is used for indoor environments to provide an alternative broadband networking infrastructure without installing dedicate network wires. They are also used in NANs.

Common wired mediums, such as fiber optical cables, are also used to construct data communication networks of the SG. Typically they offer a higher communication capacity and smaller communication delays.

Wireless networks provide the ability to eliminate the installation of physical lines. The 802.11 and ZigBee networks are the most popularly used HANs. However, it is important to notice that wireless signals are subject to transmission attenuation and environmental interference that may create some problems.

2.2 Communication in Networks

The information exchanged among devices through a network is governed by well-defined rules and conventions that is usually set out in standardized technical specifications. The Internet model, the one used by the majority of networks, divides the tasks necessary to carry out communication in several layers, where a layer leverages from the services of the layer below it. The Internet model is composed of five layers:

- Layer 5 corresponds to the application layer, used for high-level APIs and remote file access. As an example, protocols like the Hypertext Transfer Protocol (HTML) and File Transfer Protocol, are included in this layer.

- Layer 4 is the transport layer, responsible for the transmission of data between two endpoints of a network. The most common protocols used in this layer are TCP and the User Datagram Protocol (UDP)[14][15].

- Layer 3 is the network layer, which deals with inter-network connection, implementing tasks like addressing and routing. The fundamental protocol used in this layer is IP.

- Layer 2 is the data link layer, defines the type of data between adjacent network nodes. The most common used protocols are ethernet and Wi-fi.

- Finally, Layer 1 is the physical layer, responsible for the transmission of unstructured raw data on a physical medium.

2.2.1 Routing & Forwarding

The role of the network layer is to move packets from a sending host to a receiving host. To do so, two functions are necessary to do [16] :

1. Routing - refers to the network-wide process that determines the end-to-end path that packets take from the source to the destination.

2. Forwarding - refers to the router-local action of transferring a packet from an input link interface (or port) to the appropriate output link interface.

In packet switching networks, routing directs packet forwarding through intermediate nodes. Those nodes might be for example routers, gateways, firewalls, and switches. Every router has a routing table. The construction of this table is crucial and it varies according to the algorithm used. Usually routing algorithms employ the shortest path from one source to each destination, but there are alternatives that use multipath techniques that enable the use of several alternative paths.

2.2.2 IP

IP is the main communication protocol utilized in the network layer [9]. It allows the transmission of blocks of data called datagrams from sources to destinations, where the hosts are identified by fixed length addresses. Its routing function enables internetworking and essentially establishes the Internet.

IP defines packet structures that encapsulate the data to be delivered. These packets have a maximum size. This means that when a datagram is larger than the standard 1500 bytes, it will be divided into multiple packets. In addition to fragmentation IP supports the reassembly of long datagrams.

Routers communicate with one another via specially designed routing protocols, either interior gateway protocols or exterior gateway protocols, as needed by the topology of the network. IP always makes a “best effort” attempt to deliver a packet. An IP packet might be lost, delivered out of sequence, duplicated, or delayed. IP does not attempt to recover from these types of errors. The acknowledgment of delivered packets and the recovery of lost packets is the responsibility of a higher-layer protocol, such as TCP.

2.2.3 IGP

An Interior Gateway Protocol (IGP) is a layer 3 protocol used for exchanging routing information among routers within a single administrative domain [16]. This information is used to determine how packets should be transmitted while traveling through the network. There are two categories of IGP protocols: Distance-vector and Link-state.

Distance-Vector routing protocols

In a distance-vector routing protocol, a node advertises its distance value (or weight) to its neighbors. Based on this information, it is possible to compute shortest paths and fill the routing table. This process is iterative and continues until no more information is exchanged.

These protocols require that a router informs its neighbors of topology changes. Compared to link-state protocols, which require all the nodes in a network to be notified of topology changes, distance-vector routing protocols have less computational complexity and message overhead.

Distance-vector protocols normally employ the Bellman-Ford algorithm to calculate paths. An example of distance-vector routing protocol is the Routing Information Protocol [17].

Link-State routing protocols

In a link-state routing protocol each node broadcasts information about the state of its links. Globally This ensures that all routers have an identical view of the network. Each router builds a database of the network topology, aggregating the flooded network information. Then, Dijkstra's algorithm is used to compute the best path. After setting the best paths in the routing table, packets are forwarded to the destination. When a change is detected in the topology (such as when there is a link failure), the router broadcasts this change and nodes update their information in the routing table. OSPF and IS-IS are examples of link-state routing protocols.

2.2.4 Traffic Engineering

Most networks run OSPF or IS-IS routing protocols and select paths based on static link weights. The use of shortest path routing is not flexible enough to support traffic engineering in a network with a diverse set of applications. Using shortest paths conserve network resources but it may introduce problems like congestion.

OSPF Weight Optimization

The traditional OSPF routing protocol allows the use of an optimization mechanism to identify good link weight settings and thus perform traffic engineering.

In [18] the authors investigate how well OSPF routing performs on real networks, using an AT&T WorldNet backbone and synthetic inter networks. They show that finding the optimal setting of the OSPF weights is NP-hard for an arbitrary network. Considering this problem, the authors propose a local search heuristic. For the considered backbone, the heuristic allowed the computation of weight settings that made OSPF routing perform near the optimal. For the synthetic networks the results were not as good but, when compared with standard heuristics, it was possible to support a 50%-100% increase in the demand, keeping max-utilization below 100%.

The same authors in [19] present a practical approach to work within the existing framework of static link weights, without modifying the routing protocol or the routers themselves. The main point underlying this body of work is that the process of arriving at good values for the weights, or a good set of changes to the existing weight values, is handled externally from the routers. This process could depend on collected traffic measurements and topology data. The framework described has two key features: a centralized approach for setting the routing parameters and the use of link weights as the way to drive the path selection-process.

The traffic engineering approach has three main steps: measure, model and control. Selecting good link weights depends on having a timely and accurate view of the current state of the network. The operator also needs an estimate of the volume of traffic between each pair of routers. The routing model should compute a set of paths for each pair of routers. The output of the routing model can be combined with traffic demands to estimate the volume of traffic on each link, based on the topology and the IGP configuration.

MPLS

MPLS is a mechanism used for traffic engineering [10][16]. In MPLS every label switch router (LSR) makes an independent forwarding decision for the packet. Each router chooses a next hop for the packet, which is encoded in the header as a short fixed length value known as a “label” (as depicted in Figure 2.2). When a packet is forwarded to its next hop, the label is sent along with it to facilitate the forwarding decisions along the path.

MPLS requires a LDP that uses a set of procedures by which one LSR informs another of the labels bindings it has made. This protocol is used regularly to exchange labels and reachability information in order to build a complete picture of the network, which can then be used to forward packets. After full label information is exchanged, a Label Switching Path (LSP) is selected between routers. The internet resources can be allocated to multiple LSP tunnels that can be created between routers, and the traffic between the nodes is divided among the tunnels according to some policy.

In MPLS, a label can be used to represent the route. Usually in a traditional IP network each router performs an IP lookup, determines a next-hop based on its routing table, and forwards the packet to that next-hop. This process is repeated until the final destination is reached. In MPLS the first node also does a rooting lookup, but instead of finding a next-hop it searches for the final destination router. Then, the router applies a label based on this information. The following LSR uses the label to route the traffic without needing to perform any additional IP lookups. Since the label has a short fixed size, there is an improvement in forwarding speed of routers, as it avoids complex lookups. Usually at the egress router, the label is removed and the packet is forwarded normally with IP.

However, the main advantage of MPLS are the traffic management capabilities that it enables. MPLS provides the ability to control where and how traffic is routed in a network, prioritizing different services, preventing congestion and improving network resilience. In addition to LDP, MPLS uses RSVP-TE, a complex protocol with high overhead, but that supports the implementation of traffic engineering techniques.

MPLS labels can also be stacked multiple times. Some common stacking applications are: VPN/Transport services, which use an inner label to map traffic to specific interfaces, and an outer label to route through the network; and "Bypass" LSPs services, which can protect a bundle of other LSPs, to redirect traffic quickly without having to completely re-signal every LSP, in the event of a router failure.

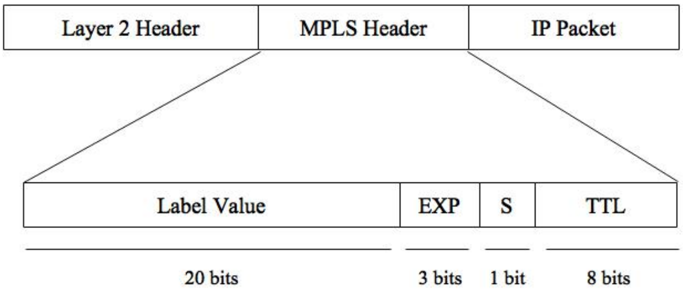


Figure 2.2: MPLS header.

2.2.5 Software-Defined Networks

Most of the success of the Internet is due to its layered protocol structure that allowed data plane services to be broken down into fundamental components and enabling innovation in each layer. In spite of this advantage, computer networks feature problems related to the complexity of the control plane, making the development and deployment of new networking features very hard because they would require a modification of all network devices.

SDN supports the decoupling of the control and the data planes. The control functionality is moved to a SDN controller, leaving the network devices with the packet forwarding functions [2]. The controller is a logically centralized platform, that runs on commodity servers. It provides the resources and abstractions to program network devices. Network applications running on top of the SDN controller implement the management services of the network (e.g., routing, link recovery and monitoring). Packet forwarding decisions are flow-based, instead of destination-based. All packets of a flow receive the same processing at the switches, i.e., they are forwarded in a similar manner.

A simplified view of the SDN architecture is presented in Figure 2.3.

- Data plane: The data plane comprises a set of network elements. It implements packet forwarding decisions that are made in the control plane.

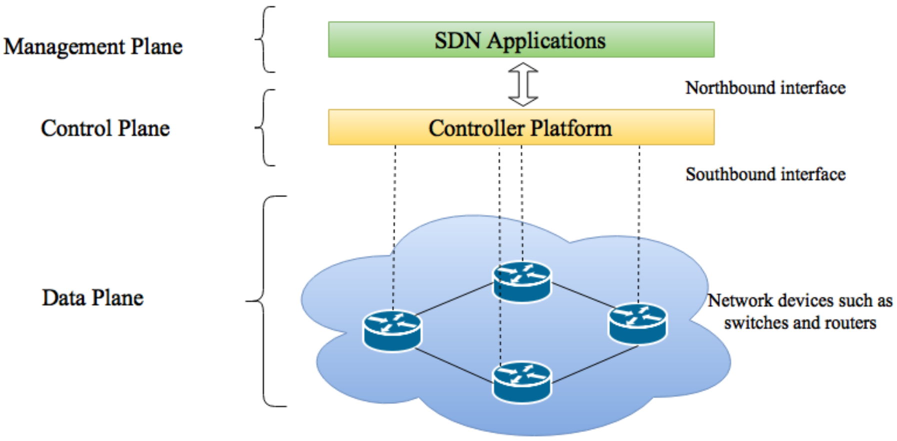


Figure 2.3: SDN Architecture. Simplified view of an SDN architecture.

- Southbound interface: In order for the data plane and the control plane to interact with each other it is necessary to define a communication protocol between the forwarding devices and the control plane elements. The most common protocol used for this task nowadays is OpenFlow [20] .

- Control plane: The SDN Controller is a logically centralized entity in charge of translating the requirements from the management plane to the data plane and providing the SDN applications with an abstract view of the network. The controller plane comprises one or more SDN controllers. Forwarding devices are programmed by the control plane using the southbound interface protocols.

- Northbound interface: Typically the northbound interface abstracts the low level instructions sets used to program the forwarding devices. This interface is offered by the controller platform through an API.

- Management plane: SDN applications are programs that programmatically communicate their network requirements and desired network behavior to the SDN Controller via the northbound interface. These applications define policies that indicate how the network should behave.

OpenFlow

Until recently, commercial network elements (e.g., switches and routers) did not provide an open interface, neither the means to virtualize either their hardware or software, which made them too inflexible. OpenFlow provides an open protocol to program the network elements [20].

OpenFlow tries to compromise generality with a degree of switch flexibility such that it is:

1. Amenable to high-performance and low-cost implementations;

2. Capable of supporting a broad range of research;

3. Ensure the isolation of flows, such as experimental traffic from production traffic;

4. Takes into consideration the vendors need for closed platforms;

An OpenFlow switch consists of at least two parts (Figure 2.4): A flow table, with an action associated with each flow entry (e.g., send to a specific port or drop a packet); and a secure channel that connects the network device to the controller, where Openflow messages can be transmitted.

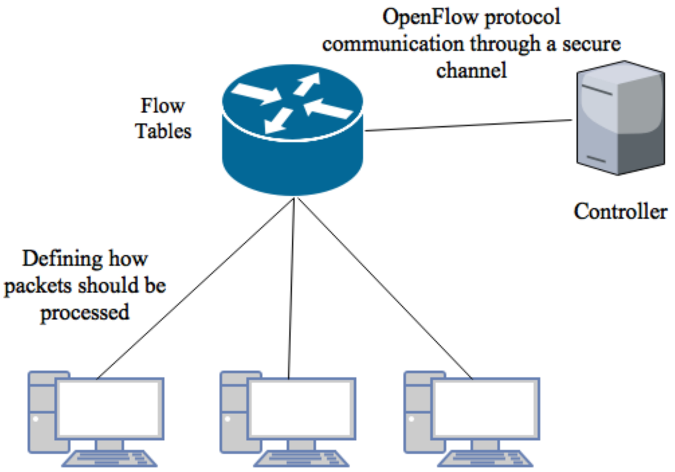

OpenFlow protocol communication through a secure channel
Flow Tables
Controller
Defining how packets should be processed

Figure 2.4: OpenFlow Switch. The flow table is controlled by a remote controller via the secure channel.

2.2.6 Traffic Engineering in SDN

Datacenter switching fabrics have enormous bandwidth demands due to the rapid increase of applications. Equal-cost multi-path (ECMP) can split a flow aggregate uniformly over a group of next-hops based on the hash of the packet header fields [21]. However, using a hash function can lead to unnecessary traffic shifts during updates. ECMP hash functions are often proprietary, making it difficult for network operators to know which traffic uses which components, complicating network troubleshooting and analysis.

The emergence of open interfaces in SDN switches suggests an alternative approach, where the controller programs the flow table to satisfy a load-balancing goal. Installing all the rules in advance may not be attractive because rules change over time and the switches have relatively limited high-speed memory. Some works attempt to solve these issues through a series of novel solutions.

Google's B4 uses SDN principles and OpenFlow to manage individual switches [21]. B4 serves multiple sites, each with a number of server clusters. Within each B4 site, the switches primarily forward traffic and do not run complex control software. The site controller layer consists of a network control server hosting both OpenFlow controllers and applications. The controller maintains network state based on information collected by the applications and switch events, and instructs switches to set forwarding table entries based on this changing network state.

Each B4 site has multiple switches with potentially hundreds of individual ports linking to remote sites. In order to scale, B4 abstracts each site into a single node with a single edge of a given capacity to every remote site. To achieve this, all traffic crossing a site-to-site edge must be evenly distributed over across all its constituent links.

The goal of traffic engineering (TE) in B4 is to share bandwidth among competing applications possibly using multiple paths. The idea is to deliver max.min fair allocation to applications. In order to do so, it associates a bandwidth function with every application. This function specifies the bandwidth allocation of the application given the flow's relative priority, which is called its fair share.

The optimal solution for the allocation of the fair share for all flow groups is expensive and does not scale well. Thus, B4 employs an algorithm that achieves similar fairness and at least 99% of the bandwidth utilization with 25x faster performance. The optimization algorithm has two main components: the tunnel group generation that allocates bandwidth to flow groups using bandwidth functions to prioritize bottleneck edges, and the tunnel group quantization that changes split ratios in each tunnel group to match the granularity supported by tables of switches.

Similar to B4, SWAN is a system that enables inter-datacenter WANs to carry more traffic using a SDN based approach [22]. SWAN achieves high efficiency while meeting policy goals, such as ensuring preferential treatment for higher-priority services and fairness among similar services. Two key aspects of the approach are global coordination of sending rates of services and centrally allocating network paths. Based on current service demands and network topology, SWAN decides how much traffic each service can send and configures the network's data plane to carry that traffic. SWAN's authors observe that it is impossible to update the network's data plane without creating congestion if all links are full. SWAN thus leaves "scratch" capacity (e.g., 10%) at each link. This enables a congestion-free plan to update the network in a series of steps.

Analysis of a production network shows that the number of rules required to fully use its capacity exceeds the limits of even the next generation SDN switches. In order to solve this problem, SWAN dynamically changes the set of paths available in the network based on traffic demand. On the same WAN, SWAN can fully use network capacity with an order of magnitude fewer rules.

SWAN supports two types of traffic sharing policies. First, it supports a small number of priority classes (e.g., Interactive > Elastic > Background) and allocates bandwidth in strict precedence across these classes, while preferring shorter paths for higher classes. Second, within a class, SWAN allocates bandwidth in a max-min fair manner.

Niagara [23] is an efficient traffic-splitting algorithm that computes OpenFlow rules to minimize traffic imbalance (the fraction of traffic sent to the “wrong” next-hop, based on weights), subject to flow-table size constraints. Niagara scales to tens of thousands of aggregates and hundreds of next hops with a small imbalance. It also provides accurate traffic-splitting with limited wildcard rules, since load-balancing weights are approximated accurately. Niagara packs rules for multiple flow aggregates into a single table and allows sharing of rules across multiple aggregates with similar weights thus improving the usage of the flow table space. Niagara also computes incremental changes to the rules to reduce churn and traffic imbalance.

2.2.7 Segment Routing

SR is an approach to improve traffic engineering and online route selection in IP networks, while preserving compatibility with the traditional MPLS data plane [5]. SR relies on label stacking to steer traffic using a source routing paradigm. With source routing, operators can specify a path from network ingress to egress using a forwarding path that is independent of the shortest path determined by the IGP. A segment is encoded as a MPLS label that is inserted in a stack called the SL. Each packet is then forwarded along the shortest path toward the network element identified at the top of the SL. Considering that segments behave just like MPLS labels, the actions push, pop and swap can be applied to modify the entries in the SL.

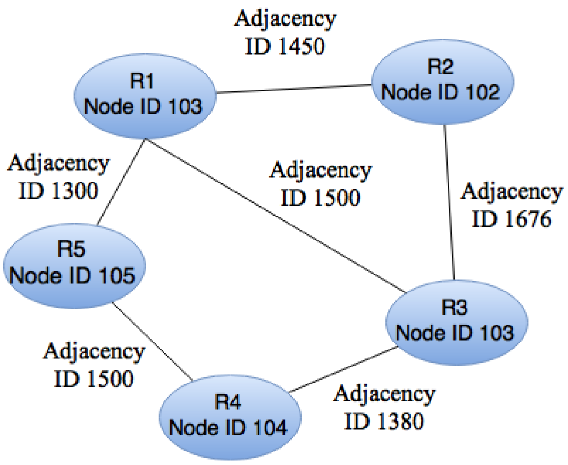


Figure 2.5: Topology using Segment Routing. Topology with node-id in each router and adjacency-id in each link. As it is possible to observe, that a topology can have two adjacency-id with the same value (link R1-R3 and link R4-R5) since they are just locally unique

Segments can be of two types, Node-ID and Adjacency-ID (depicted in Figure 2.5. Each node in the network has an associated unique Node-ID Segment identifier. Adjacency-ID identifies the links connected to a node and its neighbors (or a specific set of links).

While Node-ID are globally unique, Adjacency-ID are locally unique allowing the same Adjacency-ID to represent different links (depicted in Figure 2.6). Adjacency-ID can be globally unique if necessary, but this increases the state stored in the routers. In addition, Adjacency-ID enforces the detour of a packet through a particular interface or set of interfaces. This is key to theoretically prove that any path can be expressed as a list of segments.

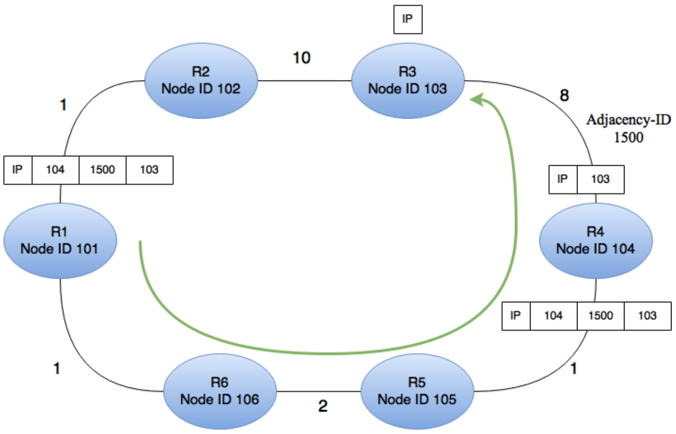


Figure 2.6: Fowarding a Segment Routing packet. It starts by forwarding the packet to R4. When arrives at R4 it pops the top segment. Then it inspect the SL again a observer that 1500 is an adjacency-id. This will force the packet through this specific link. When arrives ate R3 it pop the final label a proceeds to forward it to its destination

SR uses IGP (IS-IS or OSPF) extensions to distribute segments without a separate protocol such as LDP or RSVP-TE. Thus, scalability of transit nodes is greatly improved, since MPLS LSPs state information is not required. Unlike MPLS, there is no need to maintain path state in segment routing except on the ingress node, since packets are routed based on the list of segments that they carry. A example is displayed in Figure 2.7. In MPLS, the first LSR does a routing lookup, inserts the label for the next hop. Then at each LSR, it swaps the label for the next hop. In SR only the first router inserts the segment that will be maintained until it reaches its destination.

Segment routing can be integrated with SDN. The SDN controller knows the network topology and the current state information. Using this information, the SDN controller can pre-calculate multiple paths for load balancing. This provides a much more scalable and simple solution for traffic engineering.

Fast Reroute and Load Balancing in Segment Routing

We resort to SR to improve the resilience of the SG. This is due to its capabilities of traffic engineering, maintaining the same data plane and simplifying the control plane. Below, we give an overview of several techniques that use SR to provide fast rerouting as well as load balancing solutions.

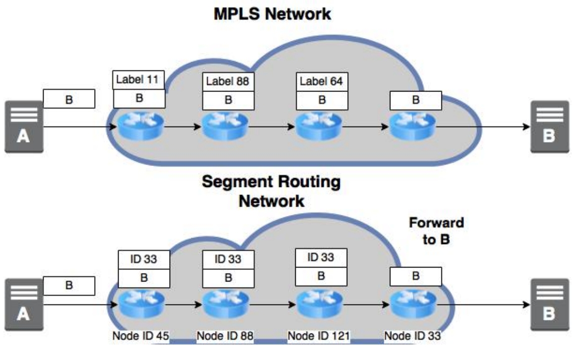


Figure 2.7: Packet labeling and forwarding in MPLS vs Segment Routing

The authors in [24] argue that is possible to improve network behavior by moving away from shortest path routing and employ SR. They propose 2-segment routing, where the traffic between the ingress and the egress router passes through exactly one intermediate node. The key decision is how to pick the most appropriate segments for each flow in order to minimize overall network congestion.

The authors consider three versions of this problem: Traffic Matrix Aware Segment Routing - the traffic matrix is assumed to be known and the objective is to determine the traffic split across different segments for each source destination pair; Traffic Matrix Oblivious Segment Routing - the traffic matrix is not known but the traffic splits are designed such that traffic is properly distributed for a wide range of traffic matrices; Online Segment Routing - connection requests arrive into the system one at a time to a SDN controller that determines which segments are picked in order to keep the load balanced.

Their solution is made from multiple linear programming formulations that have the objective to minimize the maximum link utilization. The Traffic Matrix Aware Segment Routing is easier to calculate due to the knowledge of the traffic matrix. However, the authors argue that often one does not have the traffic matrix. So, for Traffic Matrix Oblivious Segment Routing, they use an approach based on game theoretic techniques to derive the traffic matrix. This approach was used for deriving routing parameters. Online Segment Routing, besides routing the traffic, has the second objective of rejecting as few requests as possible. This is only feasible if the links are made as less congested as possible, and a solution can be obtained from a linear programming formulation.

According to their simulation results, Traffic Oblivious Segment shows that most benefits of path diversity can be captured by 2-Segment Routing. It is also demonstrated to be better than shortest path routing by a factor of 2. Online Segment Routing also performs consistently better than shortest path routing with a lower demand rejection rate.

The authors in [25] also use SR for traffic engineering. They assume that the SDN controller is requested to allocate a set of traffic flows with a specified bit rate, knowing the link capacity. They use a module with TE and SR that first allocates hop-by-hop TE paths by solving a classical flow assignment problem. Then, for each TE path, they compute a corresponding SR path to instruct the forwarding of packets along the assigned TE path.

For flow assignment, the authors implemented a modified version of an heuristic that tries to minimize the overall network crossing time. The heuristic is divided in two phases: a constrained shortest path first (CSPF) phase, where they allocate the first flows; and a heuristic re-assignment phase, which tries to re-allocate all admitted flows one-by-one, in order to minimize the global network crossing time. The second phase is executed multiple times until no more improvements are achieved. The SR assignment algorithm is then performed with the objective of finding the minimal-length SR paths corresponding to each TE path.

In [26] the authors propose a segment routing procedure to dynamically recover traffic flows disrupted by link or node failures. A central controller is used to optimize the traffic flows on the faulted topology to avoid link congestion. Their results demonstrate that, in most of the cases, traffic recovery is performed by inserting no more than two new labels in the segment list. Considering that SR natively implements ECMP aware routing, in case of multiple shortest paths toward the destination, the traffic is load balanced on a per-flow basis. In order to avoid it, it is necessary to use a more strict path, where additional labels are required in the segment list.

To implement fast recovery in SR networks, the authors propose a procedure that does not involve the controller in the traffic recovery upon failure occurrence. The forwarding table of each network node is properly configured during the network initialization phase so that when a node physically detects a failure of a connected link it is able to deviate the traffic to a backup path. The backup path calculation is made for two different types of failures: link failures and node failures. For link failures, the traffic flows are re-routed from the node detecting the failure up to the node indicated in the next label of the segment list. Basically the backup path is pre-computed excluding the supposed disrupted link from the network topology. When a node detects a failure, it checks its table and inserts more segments if needed at the top of the segment list. Without them it is possible to obtain loops. For node failures the difference is in the assumed disruption for the precomputation. In this case, the links towards the failed node are also excluded from the network.

The same authors in [27] provide an enhancement to their previous work. Even though the majority of their results require no more than two segments, there are still cases where the segment lists are too long. The enhancement requires a primary forwarding table in each node and a number of failover forwarding tables (i.e., one failover table is required for each interface of the node). When a link fails, the backup action in the primary table are executed. Following the backup actions the node first pops all the segments in the segment list except the bottom segment of the stack, and proceeds to the respective failover table. The segments inserted in the failover table will follow the shortest path without excluding the failed link. The authors also suggest that this enhancement can also be made for node failures. It would be sufficient to initialize the failover tables by computing the paths without all the links connected to the failed node. Their results show a decrease in the average segment list depth when compared with [26].

Comparing our algorithm with [26][27], we incentive the use of ECMP and LFA techniques in order to reduce the amount of segments necessary. This is fundamental since we try to minimize the number of segments used to tolerate a link failure. In average we will try to use less than 1 segment. However, the main idea of precomputing path to obtain fast reroute is the same.

2.3 Resilient Routing

The Internet is a ubiquitous platform used for a wide range of everyday communication activities. Thanks to its success, more critical services are being deployed in IP networks every day, requiring higher levels of reliability in packet delivery. Unfortunately, a simple disruption of a link in a central part of a network has the potential to affect hundreds of thousands of connections. Routing protocols like OSPF and IS-IS solve this sort of issues in a reactive manner: They rely on network-wide link state advertisements to discover network topology changes and reroute around failures. Therefore, failure recovery can take many seconds. The following works try to solve these issues through novel approaches.

In [28] the authors propose an integrated solution with simpler routers that balances load effectively under a range of failure scenarios. The arguments used is that traffic engineering and failure recovery can be achieved by the same underlying approach, by dynamically re-balancing traffic across diverse end-to-end paths in response to individual failure events, reducing routers complexity. This network architecture has three key features:

- *Precomputed multipath routing* - traffic between each pair of edge routers is split over multiple paths that are configured in advance by a management system;
- *Path-level failure detection* - failure recovery based only on the paths that have failed. A minimalist control plane performs path-level failure detection and notification which will lead to simpler and cheaper routers;
- *Local adaptation to path failures* - Upon detection of path failures, the ingress router rebalances the traffic on the remaining paths, based only on the failed path. The management system makes network wide decisions based on the expected traffic, network topology and the group of links that can fail together.

This architecture uses simple, cheap routers to balance load before, during and after failures by placing most functionality in a management system that performs offline optimizations. The management system computes these paths based on traffic engineering and failure recovery goals, and installs them in the underlying network.

The authors propose two different ways for the routers to split traffic over the working path - state-independent splitting and state-dependent splitting. In state-dependent splitting, each ingress router has a separate configuration entry with path-splitting weights for each combination of path failures to a particular egress router. State independent splitting simplifies the router configuration by having a single set of weights across all failure scenarios. So, an ingress router with three paths to an egress router would have only three weights, one for each path.

A heuristic is used to compute multiple diverse paths that ensure good load balance. This guarantees that the paths are sufficiently diverse to ensure traffic delivery in all failure states, while making efficient use of network resources.

As in the previous solution, Multiple Routing Configurations (MRC) uses a set of backup configurations based on network graph link weights [29]. This is a proactive and local protection mechanism that allows recovery in the range of milliseconds. The link weights in these backup configurations are manipulated so that for each link and node failure, the node that detects the failure can safely forward the incoming packets towards the destination on an alternate link. The backup configurations differ from the normal routing configuration in that link weights are set so as to avoid routing traffic in certain parts of the network.

The MRC approach works in three steps. First, it creates a set of backup configurations, so that every network component is excluded from packet forwarding in one configuration. Second, for each configuration, a standard routing algorithm like OSPF is used to calculate configuration specific shortest paths and create forwarding tables in each router. This provides loop-free forwarding within one configuration. Finally, a forwarding process takes advantage of the backup configurations to provide fast recovery from a component failure.

The base idea of using backup configurations prevents that even when a failure occurs, the nodes that detect it can forward traffic through another path. This is a fundamental idea to use in our work for fast reroute and recovery.

Failure Insensitive routing (FIR) ensures high service availability and reliability without changing the conventional destination-based forwarding paradigm [30]. It employs two key ideas: interface-specific forwarding and local rerouting.

Under FIR, when a link fails, the adjacent node suppresses global advertising and instead initiates local rerouting of packets, using a backup table that sets packets to be forwarded through the failed link. The nodes on the new path infer the failure from the packets in flight. When a packet arrives at a node through an unusual interface, the node can infer that probably there were failed links. These interface specific forwarding tables can be precomputed since inference about failed links can be made in advance. Thus, when a link fails, only the adjacent nodes reroute packets and all other nodes simply forward packets according to their precomputed interface specific forwarding tables without being explicitly aware of the failure.

FIR provides near-continuous forwarding of packets despite failures and improves service availability without jeopardizing routing stability. FIR requires minimal changes to conventional routing and forwarding planes. Besides using a backup table, the ability of infering errors in the networks might prove usefull in future work. However their specfic forwarding tables might require more space than the ones provided in TCAM's, specially for large networks.

Keep Forwarding [31] (KF) is designed to provide effective failure resilience for the general k-link failure case, that tolerates k failed links in a network. It is also built upon a new network model called Partial Structural Network (PSN). This approach provides flexible resilience for single and multi-failure cases, and fast recovery.

PSN is a model proposed for network routing based on directed acyclic graphs, which utilizes all links but only set directions to determined links. In the PSN model, selected links are hold undirected to achieve a better resilience. This way is PSN offers higher flexibility in failure protection.

The goal of the k-failure resilient routing problem is to maximize reachability over all failure cases. In order to improve performance while guaranteeing simplicity, KF uses an Inport-Aware Routing, where both the destination IP and the ingress port are employed for the routing lookup.

The pre-computation phase includes three steps. First, a PSN is built for each destination. Afterwards, in each PSN, every link is set with a priority. Finally, the routing table is generated based on the priority. when the building the PSN it is necessary to calculate the node weight, which is done based on the distance to the destination. Nodes with the same weight are grouped into an A-layer and the links within them are called A- links. Links between two A-layers are named M-links that consist of two types, according to the direction. To establish priority, it is considered that a router with more outgoing links should have higher potential to reach the destination, and therefore is given higher priority.

In Difane, when a switch does not know where to send a package, it transmits the packet to a special intermediate node in the network that knows a path to the respective destination [32]. The authors argue that it is necessary to install the appropriate rules in the switches, both to avoid a bottleneck at the controller and to keep all traffic in the data plane for better performance and scalability.

The key challenge is to determine the appropriate division of labor between the controller and the underlying switches, to support high-level policies in a scalable way. While the controller should generate the rules, the authors do not think that the controller should be involved in the real-time handling of data packets. In DIFANE the controller distributes the rules across the switches, called “authority switches”, to scale to large topologies with many rules. The controller runs a partitioning algorithm that divides the rules evenly and minimizes fragmentation of the rules across multiple authority switches. The space of rules is partitioned to reduce the number of rules each component must handle and enable simpler techniques for maintaining consistency.

DIFANE makes it cheap for switches to forward all data packets in the data plane, by directing “miss” packets through an intermediate switch. Transferring packets in the data plane through a slightly longer path is much faster than handling packets in the control plane. Also, it supports wildcard rules, which allow switches to have fewer rules in the TCAM. Although these techniques are interesting, they may require a specific type of topology and the addition of more switches to the network. However, the necessity to install the appropriate number of rules shows that one also has to think about the amount of rules generated.

Resilient Routing Reconfiguration (R3) [33] is a routing protection scheme that is congestion-free (all traffic demands are routed without creating any link overload), efficient in terms of router processing and robust to traffic variations and topology failures. The main idea of R3 is the use of a novel technique for covering all possible failure scenarios with a compact set of linear constraints on the amounts of traffic that should be rerouted.

A failed link is always upper bounded by the capacity of the link itself. Therefore, by creating a virtual demand for every link in the network and by taking the convex combination of all demands, it is possible to cover the entire space of rerouted traffic up to a certain bound of link failures. Through the sum of the actual demand and the set of virtual demands calculated it is possible to convert topology uncertainty into traffic uncertainty, which is easier to deal with.

R3 uses an offline precomputation phase and an online reconfiguration phase. The first one is where the routing is computed to minimize the maximum link utilization on the original network topology over the actual demand plus the virtual demand. In the second phase, R3 responds to failures using a simple rescaling procedure that does not traverse any failed link and thus can be used to reroute traffic from the failed links.

2.3.1 IP and MPLS Fast Reroute

Since IP and MPLS are appropriate protocols for the Smart Grid, we investigated their ability to perform fast reroute.

IP Fast Reroute techniques (IPFRR) work as follows [34][35]: Once a failure has been detected, traffic that previously traversed the failure link needs to be transmitted over one or more repair paths. Alternative paths are pre-calculate for all possible failures and made available for invocation with minimal delay. Several categories of repair paths are considered: ECMP, corresponding to equal cost paths; loop-free alternate (LFA) paths; and multi-hop repair paths. When there is no feasible loop-free alternate path it may still be possible to locate a router, which is more than one hop away from the router adjacent to the failure, and take advantage of it to recover from a failure.

ECMP and loop-free alternate paths offer the simplest repair paths. It is anticipated that around 80% of failures can be repaired using these basic methods alone. Multi-hop repair paths are more complex, both in the computations required to determine their existence, and in the mechanisms required to invoke them. These mechanisms are: approaches where one or more alternate Forwarding Information Bases (FIB) are pre-computed in all routers, and the repaired packets are instructed to be forwarded using a "repair FIB" by some method of per-packet signaling such as detecting a "U-turn"; approaches similar to source routing that are invoked using a normal FIB such as a tunnel; and approaches employing special addresses or labels that are installed in the FIBs of all routers, with routes pre-computed to avoid certain components of the network.

Tunnel-based approaches can use a technique called directed forwarding. It uses an IP tunneling encapsulation, or it may use a single MPLS label stack entry interposed between the IP tunnel header and the packet being repaired.

The coverage provided by multi-hop repair paths is higher, and in some topologies it is even possible to obtain full coverage. However, there is a trade-off between minimizing the number of repair paths to be computed, and minimizing the overheads incurred in using higher-order multi-hop repair paths for destinations.

The available approaches normally assume that all routers in the network are capable of acting as IP fast reroute routers, performing such tasks as tunnel termination and direct forwarding. However, this is unlikely to be the case, in part because of the heterogeneous nature of networks, and also due to the need to progressively deploy such capability. IPFRR needs to support some form of capability announcement, and the algorithms need to take these capabilities into account when calculating their path repair strategies.

As mentioned previously, MPLS uses an extension to RSVP to establish backup LSP tunnels to repair LSP tunnels [36]. There are two methods used for local protection: a one-to-one backup method where a point of local repair (PLR) (usually a node next to a failed link) computes a separate backup LSP, called a detour LSP, for each LSP that the PLR protects; and the facility backup method, where the PLR creates a single bypass tunnel that can be used to protect multiple LSPs.

In the one-to-one backup method, a label-switched path is established that intersects the original LSP somewhere downstream of the point of link or node failure. A separate backup LSP is established for each LSP that is backed up. To protect an LSP that traverses N nodes fully, there could be as many as (N - 1) detours possibly causing scalability issues.

The facility backup method takes advantage of the MPLS label stack. Instead of creating a separate LSP for every backed-up LSP, a single LSP is created that serves to back up a set of LSPs. This LSP tunnel is called a bypass tunnel. The bypass tunnel must intersect the path of the original LSPs somewhere downstream of the PLR.

RSVP-TE Fast Reroute methods allows full network coverage but with a high complexity in terms of operation, as well as potential scaling issues [37]. RSVP-TE inherited some properties of RSVP that adversely affect its control plane scalability. In particular, reliance on periodic refreshes for state synchronization between RSVP neighbors and for recovery from lost RSVP messages, reliance on refresh timeout for stale state cleanup, and lack of any mechanisms by which a receiver of RSVP messages can apply back pressure to the sender of these messages. In addition, when RSVP-TE is used for the MPLS control plane, the path state is held at every router through which the LSP passes, including head-end, tail-end and any intermediate (mid-point) routers.

2.4 Final Considerations

In our work, we employed some of the previously proposed approaches and integrated them in a solution to recover from a single link failure using SR in a SDN network that targets SG. In our solution we aim to provide fast reroute as the main goal while doing traffic engineering. Thus, we allow the network to recover quickly after a failure without congesting it.

Chapter 3 - Design and Implementation

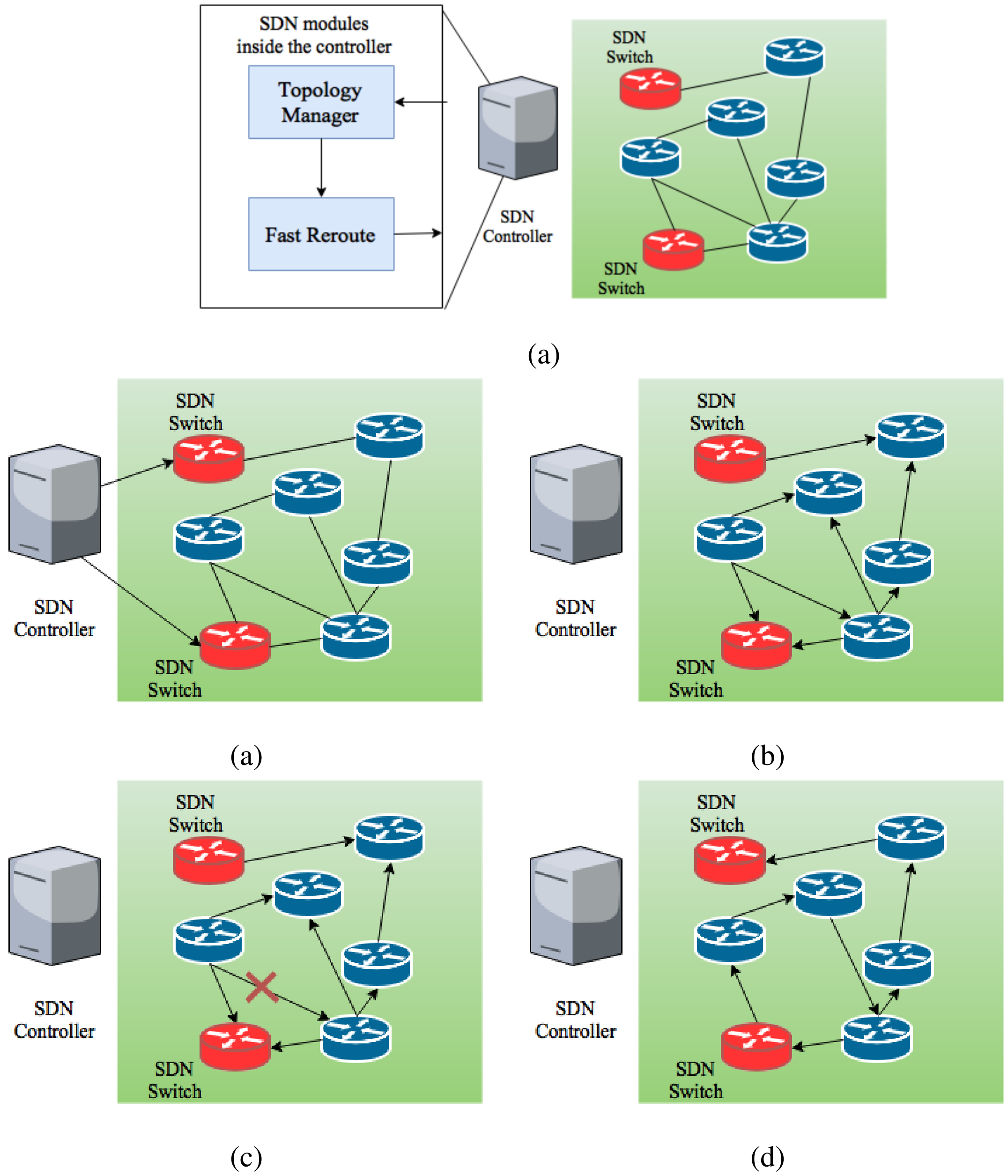
It is common today to switches have the capability to detect failures that affect the links directly connected to its ports. Detection is made almost immediately for ordinary forms of the problem, such as when a cable is broken. This functionality can be exploited to restore network connectivity by diverting the traffic, which would be transmitted through the failed link, to the remaining ports that are still operational. Based on this idea, the main objective of the protocol for fast reroute proposed in this thesis is to recover the network from a single link failure while guaranteeing maximum coverage, minimum overheads and loop freeness. In order to accomplish the properties mentioned above, we designed a module to run in an SDN controller that pre-computes backup paths and installs them on routers. We follow a SR approach in our design. Finnaly, we address how the algorithm was developed through an UML diagram, and add few considerations

3.1 Algorithm for Fast Reroute

In Figure 3.1 the fast reroute module obtains an up-to-date view of the network topology by querying the topology manager. Based on the topology, it computes for each link a path that detours packets in case of failure, allowing the network to recover quickly. After executing the algorithm it sends the information to the edge switches (SDN switches) that will distribute the backup path to the remaining switches. These alternative paths are configured in the switches in backup flow tables that are used only when a port becomes disconnected (i.e., the switch detects locally the failure of the link connected to the port). Therefore, recovery can be accomplished without the need to wait for the routing protocols to converge or for the SDN controller to intervene, allowing the traffic to continue to flow without loops.

The algorithm assumes switches to employ ECMP, LFA techniques [34], including LFA, Remote LFA (RLFA) and Direct LFA (DLFA), and SR. A LFA is an alternate next hop through which packets can be sent in case of failure without creating any loops. We follow a SR approach, an emerging technology for IP/MPLS networks that provides the ability to source-route. With source routing, operators can specify a path from ingress to egress that is independent of the shortest path determined by the IGP. As mentioned previously, SR uses segments instead of labels, such as Node-ID Segment that identify a node in the network and Adjacency-ID that identifies the links connected to a node. Segments are added to the SL in the ingress node and forwarded through the network to the element at head of the list.

The algorithm is executed in a number of steps, which are attempted in order until a recovery solution is found (see Table 3.1 and Algorithm 1). The several steps of the algorithm are explained next.



(a)

(a)

(b)

(c)

(d)

Figure 3.1: The several steps of a fast reroute mechanism: (a) shows the computation (b) installation of normal and backup rules on the edge switches;(b) Using the normal forwarding rules. (c) displays a failure in a link; and finally (d) fast rerouting after detecting the failure, allowing the traffic to flow.

Notation used in the algorithm	
N	Set of all nodes in the network $\{n_1, n_2, ..., n_m\}$
NB_n	Set of all neighbor nodes of node n $\{nb_1, nb_2, ..., nb_n\}$
$dist(n, d)$	Distance from node n to node d through shortest path routing
$dist_f(n, d)$	Distance from node n to node d through shortest path after a failure f
$obtainNB(n)$	Obtain the set of neighbor nodes of node n
$PSpace(n)$	Calculate the P-Space and Extended P-Space of node n
$QSpace(n)$	Calculate the Q-Space of node n
$intersect(P, Q)$	Intersects two sets
$segmentSelect()$	Select a specific segment
$2segmentSelect()$	Selects 2 specific segments

Table 3.1: Functions and sets used in the algorithm.

Algorithm 1 Algorithm

```
for each node n in N do
    for each destination d in N do
        for each possible link failure f of node n do
            Step 1: ECMP
            if $dist(n, d) == dist_f(n, d)$ then
                no segment required
                continue to next iteration
            Step 2: LFA
            NB= $obtainNB(n)$
            for each neighbor nb in NB do
                if $dist(nb, d) < dist(nb, n) + dist(n, d)$ then
                    no segment required
                    forward to nb with the shortest path to d
                    continue to next iteration
            Step 3: RLFA
            P = PSpace(n)
            Q = QSpace(d)
            PQ = $intersect(P, Q)$
            if PQ is not empty then
                $segmentSelect()$ of PSpace or Extended PSpace
                add segment to SL
                continue to next iteration
            else
                Step 4: DLFA
                $2segmentSelect()$
                add the two segments to SL
                continue to next iteration
```

3.1.1 Step 1 : ECMP Safeguard

Nowadays it is common for a switch to support the use of ECMP [38]. If one of the links fails, the switch can divert the packets using the remaining equal cost paths automatically. This is valid because it is already known that the other paths are shortest path to the respective destination[35] [39]. In this case it is not necessary to insert any segment to recover from a link failure. This is possible to observe in lines 4 to 7 in Algorithm1. Consider the Figure 3.2, consider B the source of packets and G the destination. If link B-G fails, B can automatically move all traffic to route B-C-G because it has the same distance/cost to reach node G.

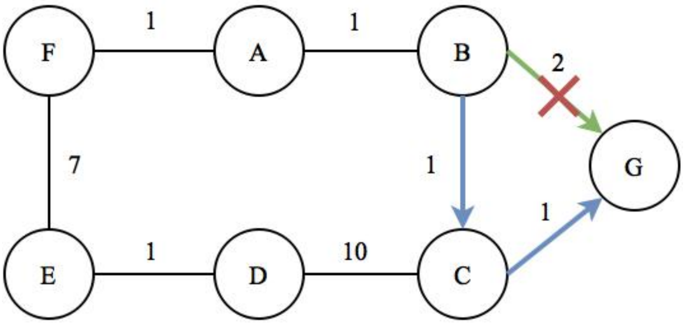


Figure 3.2: ECMP step: B wishes to send a packet to G. If link B-G fails, we can use ECMP through the path B-C-G which has the same cost.

3.1.2 Step 2: Loop Free Alternative Neighbor

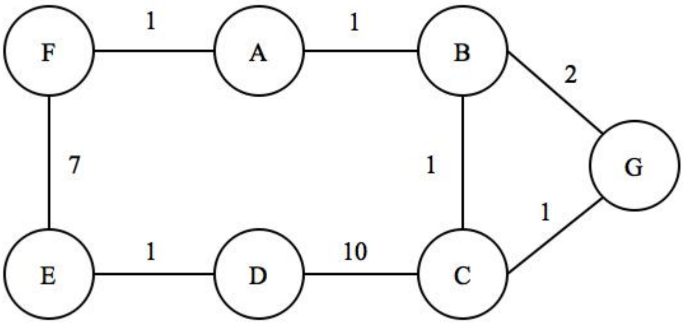


Figure 3.3: LFA step: Example of a network topology.

Consider the network depicted in Figure 3.3, further consider node B to be the sender and node D the receiver, it will use the path with lower cost $(B - A - F - E - D)$. If, however, link (A, B) fails, then B needs to find an alternative neighbor to forward the packet to D as destination [40]. ECMP cannot be applied in this case because the paths that go through the available neighbors have a higher cost than the original path (namely, 11 and 13 versus 10). In general, a solution is found without creating loops if the following condition is verified. The source (src) needs to find a neighbor (n) that is closer to the destination (dst), ensuring that i) the cost/length of the route from the neighbor to the destination is smaller than ii) the added cost/length of the neighbor to the source and from the source to the destination (i.e., the original shortest path). This relation is expressed as:

$$dist(n, dst) < dist(n, src) + dist(src, dst), \tag{3.1}$$

where, the function $dist(i, k)$ denotes the length of the shortest path from i to k on the original network topology without the link failure. One should notice that if the relation does not hold, then the neighbor will forward the packet accordingly to the shortest path, which in this case would mean returning it back to the source, thus creating a loop.

The second step of the algorithm will use the LFA equation provided in (3.1) for the respective neighbors of the source node (lines 8 to 14 in Algorithm 1). If one of the neighbors is a LFA node, then it will not be necessary to add a segment to the SL. It is just necessary to forward the packets through the respective LFA neighbor after the failure. If more than one neighbor node provides LFA coverage, the one with smaller $dst(n, dst)$ is chosen, in order to transmit packets through the shortest path possible (for simplicity, this optimization is not represented in Algorithm 1).

Returning to our example src is the source node B, dst is the node D, n is a neighbor of src other than the failed next-hop (A in the example). Neighbor C and G are for instance a LFA neighbor for B towards node D.

$$dist(C, D) < dist(C, B) + dist(B, D) \Leftrightarrow 10 < 1 + 10 \tag{3.2}$$

3.1.3 Step 3: RLFA (1-Segment)

If there is no LFA neighbor, one has to resort to other techniques to find alternative paths. Our proposal is to search for a RLFA where the source will need to create a virtual LFA tunnel to carry the packet to a node that is not a direct neighbor. This tunnel must be reachable from the source without traversing the failed link. When the tunnel exists, it has as endpoint an intermediate switch that is able to deliver the packet to the destination without looping back to the source. This intermediate node will be the segment inserted in the packet header at the source. In order to do so, we will use the P and Q-Space techniques used by RLFA [41] (lines 15 to 23 in Algorithm 1).

P-Space

The P-Space is the set of switches that can be reached from the source after the link failure, in other words, the switches where the cost to reach them is smaller than the cost to reach them through the failed link.

Consider the failure of link A-B in the topology in Figure 3.4 and that A is the source and C is the destination. The set of nodes that can be reached from A without traversing the failed link A-B is termed the *P-Space of A*. In this case, the P-Space for node A with a faulty link A-B will be $P(A) = \{F, E, D\}$.

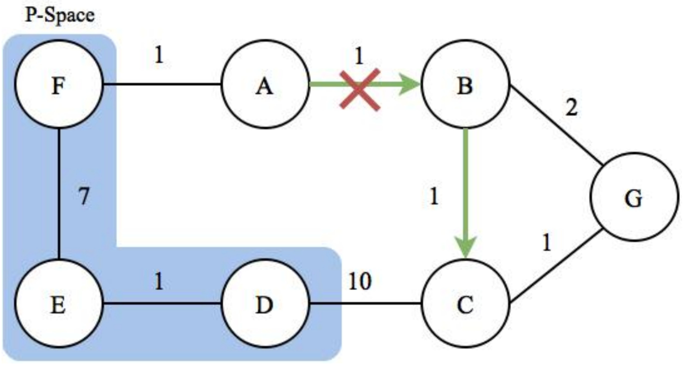


Figure 3.4: Representation of P-Space after the link failure A-B. In this example A is the source and C the destination.

Q-Space

The Q-Space is the set of nodes from which the destination can be reached, by normal forwarding, without traversing the failed link. Considering Figure 3.5 with the same case as above, the Q-space of node C will be $Q(C) = \{D, G, B\}$.

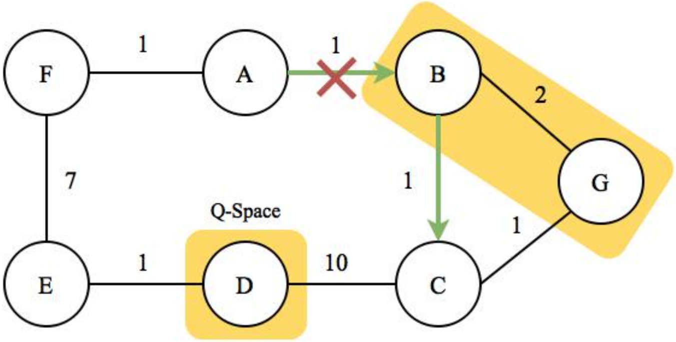


Figure 3.5: Representation of Q-Space after the link A-B failure. Where A is the source and C the destination.

PQ switches

The PQ switches are the switches in the intersection of the source P-Space with the destination Q-Space. These switches define the set of viable switches that can be added to the SL, since the source can forward to these switches without traversing the failed link while ensuring that they can transmit the packets to the destination (as they would not loop back). In the example, if $P \cap Q$ Space, the only the result $PQ = \{D\}$. In other cases, however, there might cases where PQ intersection is empty. In those cases, it is necessary to modify the solution by calculating the Extended P-Space. In some cases, the PQ intersection will have several nodes, making possible to employ different mechanisms to choose which node to select as intermediary node.

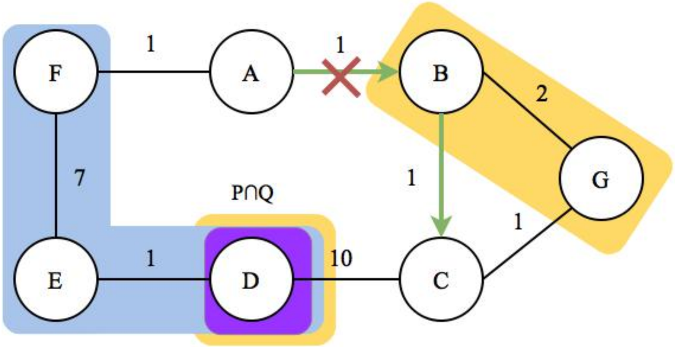


Figure 3.6: Representation of PQ-Space, in a situation where A is the sender and C the destination. The Node-ID Segment from D will be added to the SL at the A switch. As a result, the packet will be forwarded to F that will inspect the label and forward it to D. When the label is popped at D it will then forward the packets to C.

Extended P-Space

Consider now that B the sender and A the destination and that the link B-A fails in the same topology as above (represented in Figure 3.7). The P-Space of node B is $P(B) = \{C, G\}$. The Q-Space of node A is $Q(A) = \{F, E, D\}$. This case has no intersection between these two sets $P(B)$ and $Q(A)$, and therefore the forwarding solution mentioned above cannot be utilized. Consider SN as the set of nodes adjacent to the src (i.e., the neighbours) without passing through the failed link. The Extended P-space of the src is the $P(src) \cup P(n)\ \forall n \in SN$, in other words the extended P-Space is the union of the source P-Space with its neighbors P-Space. In this case, the Extended P-Space for node B is $Extended - P(B) = \{C, G, D\}$.

When using the Extended P-Space instead of the P-Space to compute the intersection with the Q-Space, there is potentially a new outcome for the PQ switches, which may no longer be empty. When this happens, one of these switches can be added to the SL (again, using a selection algorithm as the ones proposed in section 3.2), so that packets are forwarded through that switch. In the example, the PQ now includes nodes D. In case of failure of link B-A, B should forward packet with the SR label D, D pops up the label and forwards to destination A using shortest path.

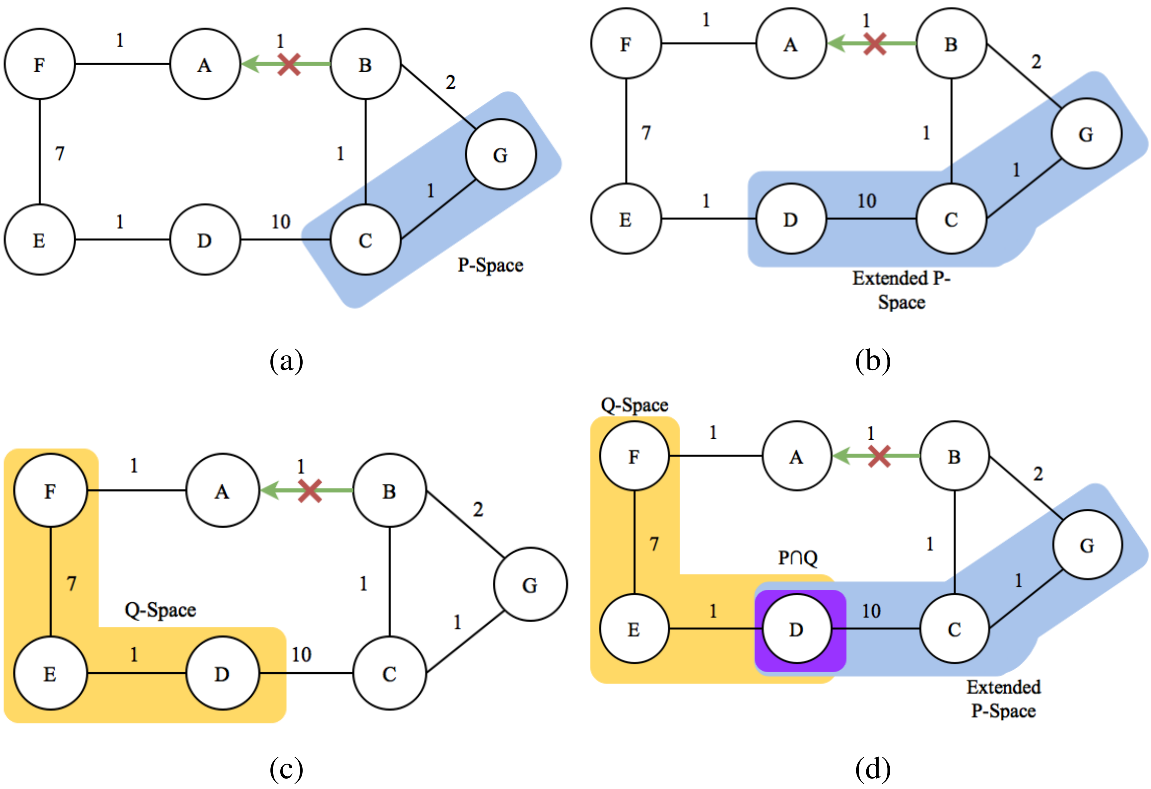


Figure 3.7: Representation of the Extended P-Space through multiple steps: (a) shows the computation of B P-Space;(b) presents the new Extended P-Space; (c) displays the Q-Space; and finally (d) shows the $P \cap Q$.

3.1.4 Step 4: DLFA (2-Segments)

If the PQ set is empty even with the Extended P-Space then it is necessary to use DLFA. DLFA finds which nodes in the Extended P-Space are adjacent to a node in the Q-Space. After discovering which nodes are adjacent it is possible to add to the SL the Node-ID corresponding to the switch in the Extended P-Space and an Adjacency-ID Segment that forces the switch to forward the packet to the switch in the Q-Space. Overall, the SL contains two segments, defining a route across the network that guarantees the packet delivery (lines 24 to 27 in Algorithm 1).

In Figure 3.8 there is an example where A wants to send information to B and link A-B fails. In this case, $P(A) = \{F, E, D\}$ and $Q(B) = \{C, G\}$. When Extended P-Space is computed, it does not add switches to the set, and therefore the intersection stays empty. If we check the switches in the P-Space that are adjacent to the Q-Space, only D is adjacent to node C. Therefore, in order to ensure delivery of packets it will be necessary to add to the SL the Node-ID Segment of D and the Adjacency-ID A_{DC}.

It is important to mention that a router must limit the amount of time an alternate next-hop is used after the primary next-hop has become unavailable, since eventually the routing tables in the network will be updated to reflect the changes in the topology.

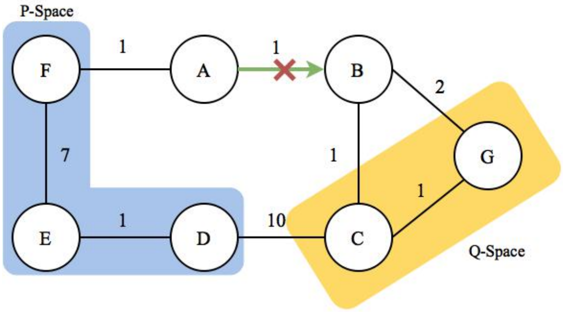


Figure 3.8: Empty intersection.

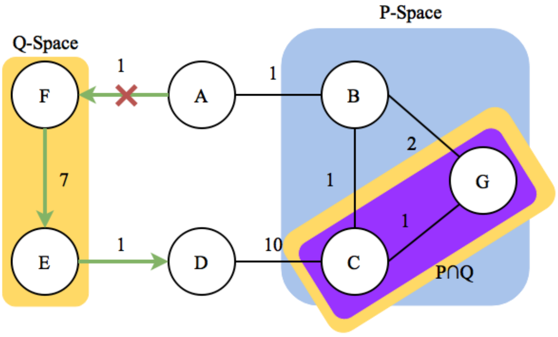


Figure 3.9: PQ Switches with multiple nodes.

This ensures that all possible transient conditions are removed and the network converges according to the deployed routing protocol. The use of the alternate next-hops for packet forwarding should terminate[35] if the new primary next-hop was loop-free prior to the topology change, or if notification of an unrelated topological change in the network is received.

3.2 Choosing from Multiple Alternative Segments

As been explained before in Step 3, if the PQ set is composed of several nodes, it is necessary to select one of them as it is shown in the algorithm through $segmentSelect()$ (line 20 in the Algorithm 1). We propose two strategies for this purpose. The first approach we used, which we called **Fast Segment Drop** (FSD), employs information from convergence. Convergence is the state of a set of switches that have the same topological information about the network in which they operate. This routing state defines the (shortest) paths that are in use at a given instant. A pre-convergence path is a path that is in use before the failure. A post-convergence path is a path that is in use after the failure. The set of pre-convergence and post-convergence paths are typically different because the routing state changes with the link failure.

Through the use of an SDN controller we know the current state of the network, and we can assume link failures to calculate new post-convergence paths. FSD obtains the post-convergence path, then checks first if there is more than one switch from the PQ in it. In the affirmative case, we choose the switch that is closer to the source providing lower overhead possible. However, if there are no PQ nodes in the post-convergence path, we simply choose one that is closer to the source. This provides slightly higher overhead but will be still lower than using two segments.

In Step 4 through $2segmentSelect()$ (line 25 in the Algorithm 1), we use a similar approach but for the set of switches in the P-Space that are adjacent to switches in the Q-Space. If it there are nodes adjacent to Q-Space that are in the post convergence path, we will choose the node closer to the source and the adjacency segment towards the Q-Space, providing once again lower overhead. However, if there is not, we verify the remaining nodes and choose the closer to the source along with the respective adjacency segment towards the Q-Space.

This approach allows us to minimize overhead. The use of the SL imposes an overhead on the network that results from the transmission and processing of packets with a larger header (because they have extra segment information). Therefore, one would like to reduce the number of segments that are included in the SL and the number of hops until they are popped out. If the segment selected are in the post convergence path it will always have the lowest overhead possible.

The second strategy we propose provides a load balancing solution, **Congestion Avoidance Segment** (CAS), where we choose the segment that is associated with a path with the smallest link utilization. After we calculate the set of PQ switches, if it is not empty, $segmentSelect()$ tests the all paths from the source to the possible intermediate switch (a switch that is in the PQ set) and then from the intermediate switch to the destination, and we choose the path with less link utilization. However, if PQ set is empty, in $2segmentSelect()$, we choose the path with less link utilization with the best node (a node in the P-Space that is adjacent to a node in the Q-Space) and its adjacency segment, inserting two segment in the SL. This solution will have greater overhead than the previous but avoids passing through highly congested links.

3.3 Implementation

All software components were implemented in Java within approximately 2700 lines of code. The machine where the experiments where performed had an Intel Core i5-4570 CPU with 3.20GHz x 4 and 4 GB of RAM memory. The software environment was Ubuntu 14.04 LTS with Java(TM) SE Runtime Environment (build 1.7.0 07-b10) 64 bits.

Figure 3.10 shows the class diagram for the software development. *SegmentRouting* is the abstract class responsible for the computation of the algorithm presented previously (Algorithm 1). It requires all shortest paths from every node to all other nodes that are obtained by class *ShortestPath*. The class *ShortestPath* computes all paths from a source node to all possible destinations through *Dijkstra* class. It also computes all paths from a source node to all possible connections for each adjacent link failed. This provides us all possible paths for all single link failures. *Dijkstra* uses instances of the class *Distance* that stores the shortest path computed by *Dijkstra* and also the sum of all weight values of the respective path. *SegmentRouting* has instances of the respective *graph* and *allSL* that is responsible for maintaining the SL for each switch. *SegmentRoutingFSD*, *SegmentRoutingRandom*, *SegmentRoutingTI-LFA*, *SegmentRoutingCAS* are the classes that implement the interface, each one have the same attributes and methods but each one implements different strategies.

SegmentList is composed by all the segments (Node-ID Segment presented in *Segment* or adjacency segments in *AdjacencySegment*) necessary to perform fast reroute after a failure (it can be empty if ECMP or LFA are enough to find a solution).

The package *Graph* is designed to build a graph with a certain topology. The class *Graph* is composed of multiple nodes and edges. The constructor of Graph defines the number of nodes added, and *Edge* class allows to set the respective weight values for links and its respective bandwidth.

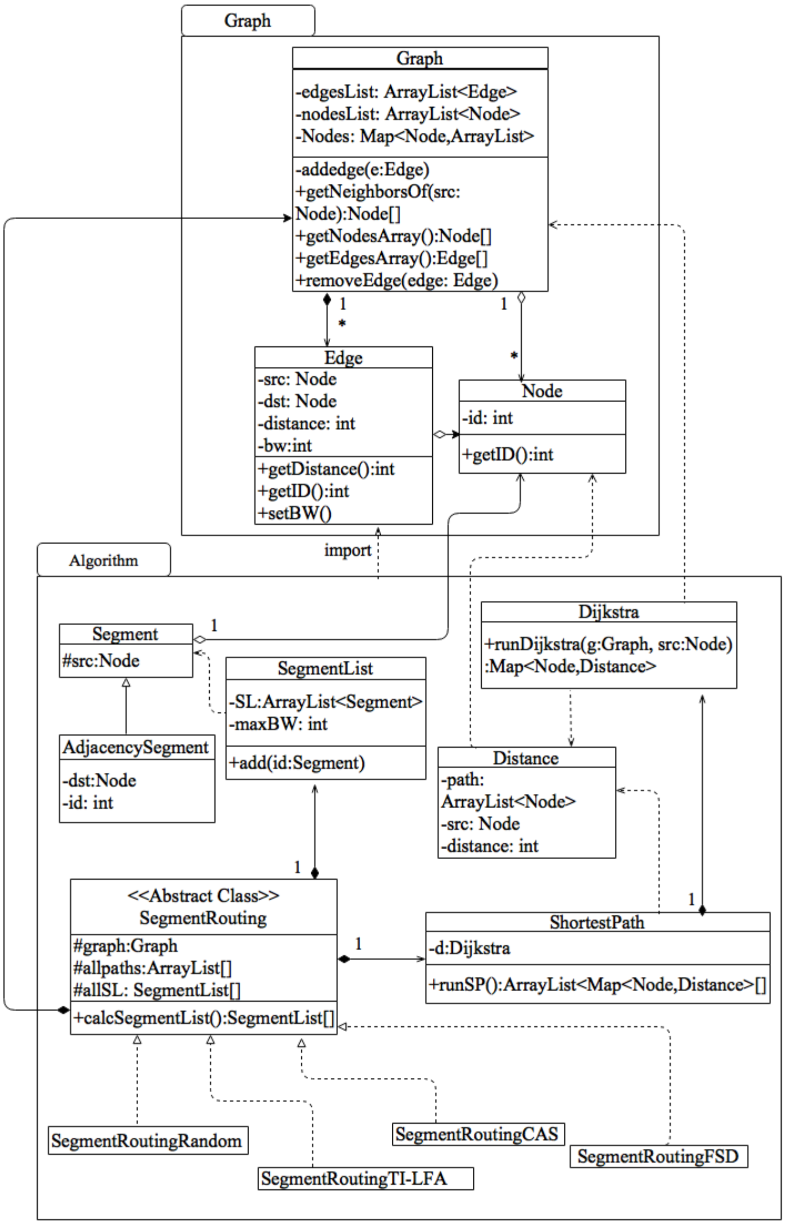


Figure 3.10: UML class diagram.

3.4 Final Considerations

We introduced in this chapter an algorithm that uses multiple steps to attempt to immediately recover the network after a link failure. We described the several steps of our algorithm, where in each step is used a different technique that ensures packet delivery. We also presented the several segment selectors, that provide different properties such as minimizing packet overhead and load balancing. Finally, it was also explained the how the algorithm was implemented its respective UML diagram.

Chapter 4 - Evaluation

To evaluate the properties of our solution, we implemented two versions of our algorithm with the two strategies mentioned in section 3.2. We evaluated our strategies against a random strategy and TI-LFA (a mechanism that will be introduced in section 4.1).

Our main goals are to analyze the packet overhead and load balancing abilities of each algorithm on several network topologies. Another goal is to compare the coverage of our proposal with the IP fast reroute and MPLS fast reroute.

We start by introducing random and TI-LFA selectors and then the environment setup, including the topologies used, link weights and bandwidth values. Finally, we report and discuss our results for each solution and selector.

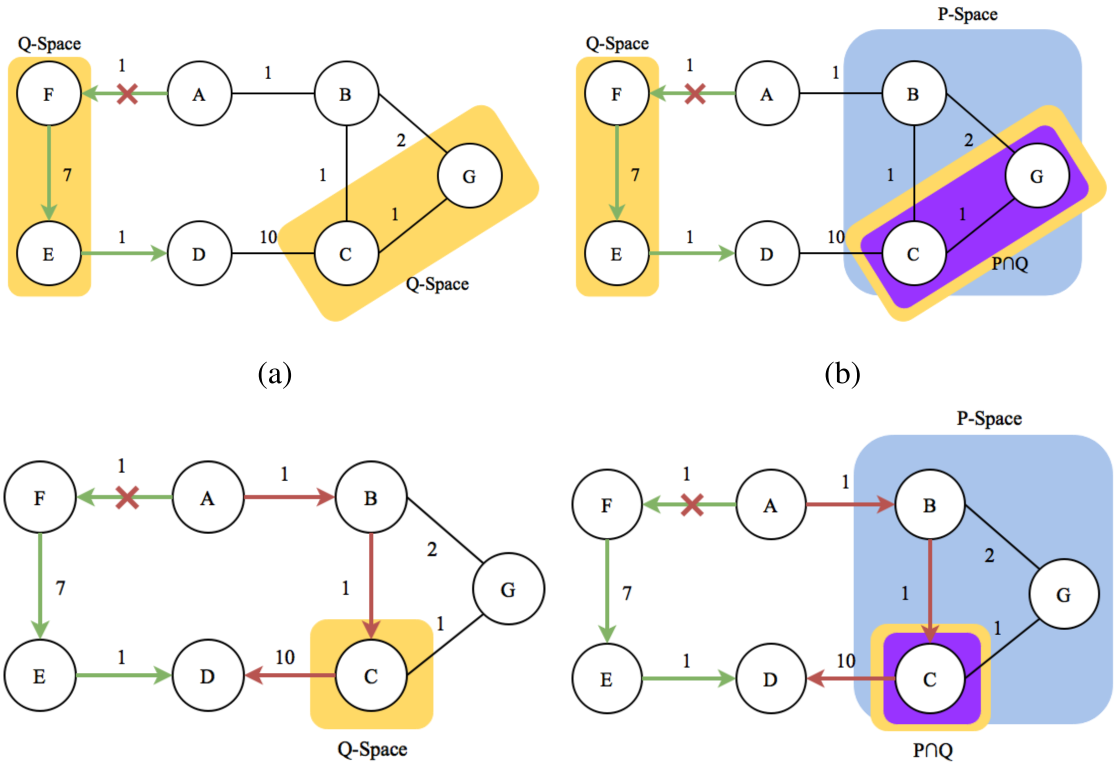
4.1 Random and TI-LFA strategies

In order to compare our strategies (FSD and CAS), we implemented a random selector and TI-LFA algorithm. After discovering the PQ node set, the random selector chooses one node belonging to the PQ set randomly that will be added to the SL. This is also used for the $2segmentSelection()$, in which we find all the possible nodes in the P-Space that are adjacent to a node in the Q-Space and choose one randomly along with the respective Adjacency-ID Segment.

In [6], the authors intend to reduce the amount of path changes and service transients, making just one transition (pre-convergence to post-convergence) instead of two (pre-convergence to fast reroute and then to post-convergence) when a failure occurs. To implement this idea, they consider the *pre-convergence path* as the shortest path between a source and destination before a link failure, and *the post-convergence path* as the shortest path between a source and destination after a failure. Then, the authors use similar techniques to ours but instead they intersect the Q-Space with the post-convergence path. Using this new Q-Space forces the repair path to go through the post-convergence path.

In Figure 4.1 we have the representations of our algorithm and TI-LFA. The P-Space is the same, however the Q-Space computed by TI-LFA only considers the nodes that are in the post-convergence (shortest) path (represented in red in Figure 4.1 (c)), in this case C. When $P \cap Q$ TI-LFA only has one node that can be chosen.

In the implementation of this solutions, we also chose to select the PQ node closer to the source since the authors do not mention which none they select. While in Fast Segment Drop we *try* to follow the shortest path, in TI-LFA we are *bound* to use the post-convergence (shortest) path. When using Fast Segment Drop there will be some cases where instead of using the shortest path the algorithm will choose a longer path with just one segment. In TI-LFA it will always use the shortest path, but it with one or more segments.



(a)

(b)

(c)

(d)

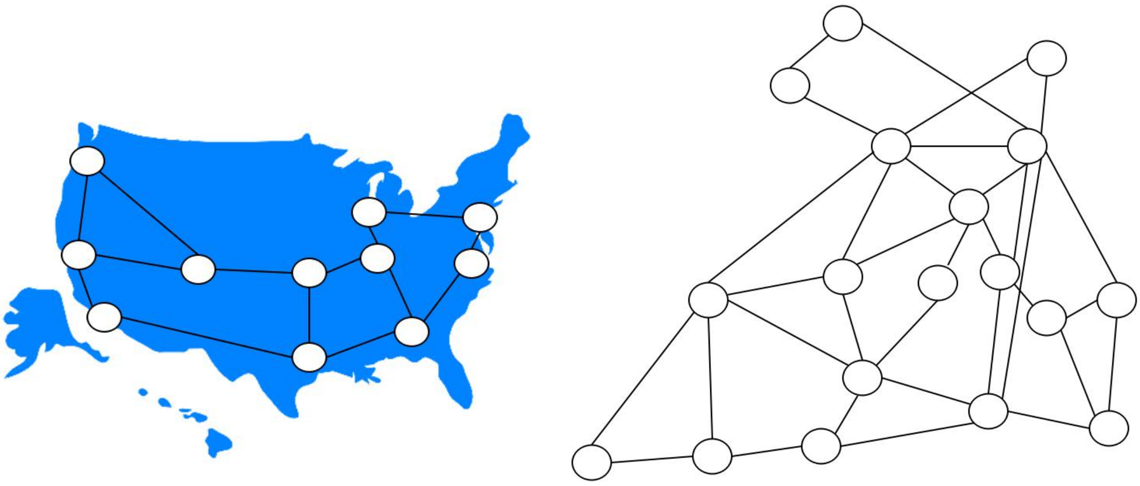
Figure 4.1: Representation of TI-LFA. The example considers the case where A is the sender and D is the destination but the link A-F fails. In (a) we can observe the several nodes in the Q-Space while in (c) there is just one node, the one that belongs to the shortest path (which is A-B-C-D). In (b) and (d) we can observe that $P \cap Q$ obtains two different results for the two versions of the algorithm.

4.2 Environment setup

To evaluate our solution we considered several topologies, divided in three groups of topologies: (i) real topologies; (ii) random topologies generated with BRITE[39]; (iii) regular topologies (i.e., grids and rings) and fat trees.

We considered the following real topologies:

- Abilene Network (Abilene), a high-performance backbone network composed mostly of universities and some corporate and affiliate institutions, across the US. In our tests we considered 11 nodes (depicted in Figure 4.2 (a)).



(a) (b)

Figure 4.2: In (a) the Abilene topology; in (b) the Pan European or GÉANT topology.

- GEANT is the pan-European data network for the research and education community (Depicted in Figure 4.2 (b)) (PanEU). It interconnects national research and education networks across Europe. The GEANT project combines a high-bandwidth, high-capacity 50,000 km network with a growing range of services. In our tests we considered 18 nodes.

- The National Science Foundation Network (NSF) [42], in Figure 4.3 (a) (NFS), is composed by 14 nodes. It was initially created to link researchers to NSF-funded supercomputing centers. Through further public funding and private industry partnerships it IS developed into a major part of the Internet backbone.

- The future smart grid connection infrastructure of EDP, a portuguese electric distribution company. This infrastructure is composed by the union of three rings (as shown in Figure 4.3 (b)). The first ring represents the core network that hosts critical application servers and acts as a gateway to other networks. The second ring is the aggregation network that has multiple terminations to aggregate traffic from the next ring, the edge. The edge enables services related to the transmission of data from smart meters, substation control and monitoring. In practice, the core is connected to multiple aggregation rings, and each aggregation ring has multiple edge rings attached, providing a huge network infrastructure. In our tests we generated two topologies based on this organization, one with 21 nodes (1 core ring, 1 aggregation ring and 1 edge ring) and another with 140 nodes (1 core ring with 10 nodes, 10 aggregation rings with 5 nodes each, 8 edge rings with 10 nodes each).

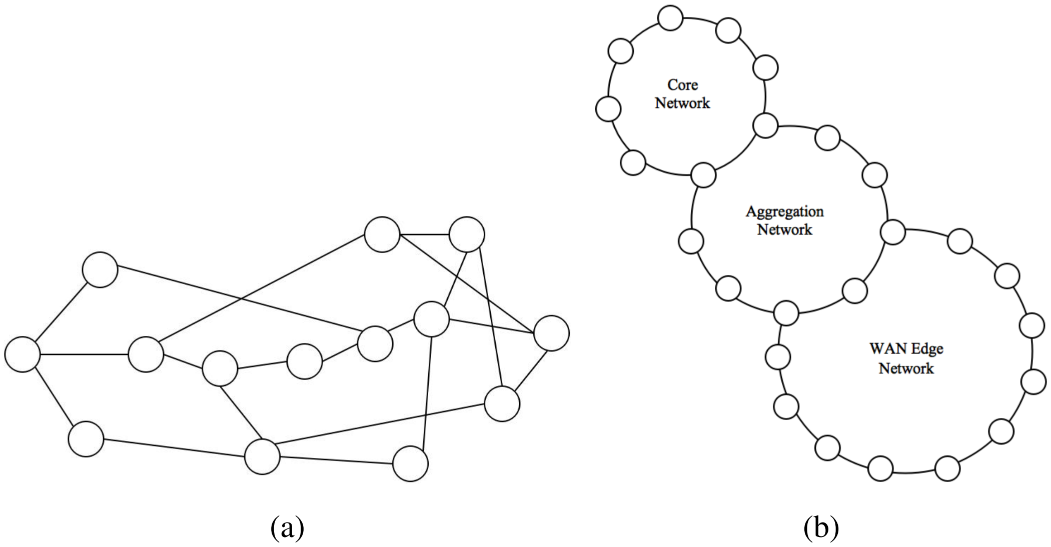


Figure 4.3: In (a) the NSF topology and in (b) EDP smartgrid topology.

In order to generate more generic topologies we used BRITE[39] a random topology generator, that employs a Flat Router-Level model where each network ID is represented individually in the routing table (depicted in Figure 4.4). The used router model is Waxman, which refers to a generation model for a random topology using Waxman's probability model for interconnecting the nodes of the topology. We generated topologies with 16, 32 and 64 nodes.

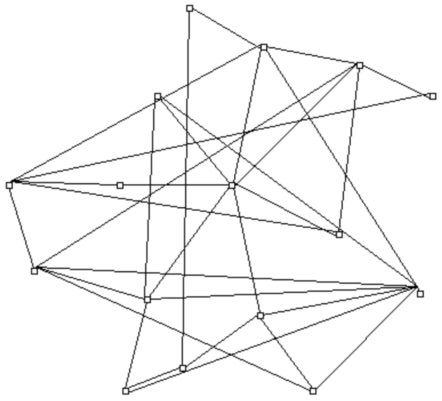


Figure 4.4: Example of a BRITE topology with 16 nodes.

We also generated regular topologies:

- Rings with 16, 25 and 49 nodes (presented in Figure 4.5 (a)).
- Grids with 4x4, 5x5 and 7x7 nodes (as depicted in Figure 4.5 (b)).

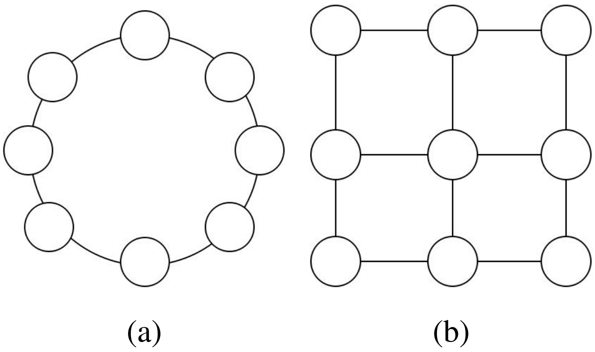


Figure 4.5: In (a) an example of a ring topology and in (b) of a Grid topology.

- Fat trees, which are topologies employed in large scale datacenters. It is divided in k pods, with each pod containing 2 layers with k/2 switches, and a core layer (as shown in Figure 4.6). The lower layer is called edge and the upper layer aggregation. Each switch in the lower layer is connect to k/2 hosts and to k/2 switches in the aggregation layer. The aggregation layer is also connected to k/2 core switches. The core layer is made of $(k/2)^2$ switches, each one is connected to k pods. The advantages of using this topology is that all switches are identical providing the ability to use commodity switches across the network, and by having multiple paths of the same length it allows multiple paths to be explored

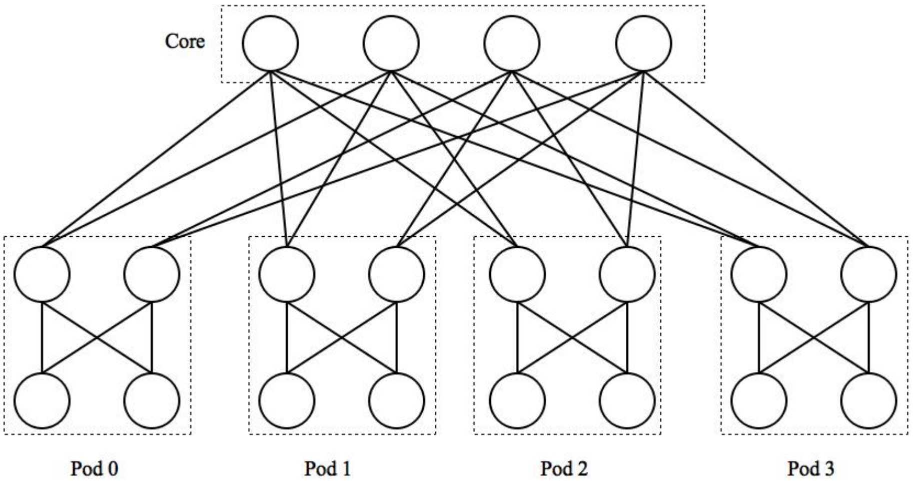


Figure 4.6: Example of a 4-ary Fat Tree topology.

For some topologies (namely EDP, grids and rings), it was not possible to obtain the link weight values and therefore we assigned values between 1 and 10 randomly. For these topologies, the algorithms were run 100 times with different random link values. In addition, for the BRITE topologies we tested the algorithms on 100 different topologies of each size. The results presented in the next section are the averages of these 100 experiments.

In order to perform congestion analysis, each link was assigned a bandwidth value. We established that the maximum link load is 100Mbps, so on average the networks have a link utilization bellow 50% (in order to tolerate failures). We used a normal distribution function where the values tend to cluster around the average. The average was set to be 30Mbps with a deviation of 20Mbps. Thus, 70% of values are between 10-50Mbps and 95% are between 1-70Mbps.

4.3 Evaluation results

This section evaluates the proposed algorithm considering different strategies: FSD, CAS, Random and TI-LFA. The objective is to:

- Analyze the average segment list size (SLS) in order to understand what strategies require more segments to tolerate a single link failure.

- Analyze the average packet overhead. To evaluate packet overhead we use the technique provided in[43], where the number of segments that exist in the SL at each hop of the transmission path is computed. A SL that contains segments all the way to the destination would have a higher overhead when compared to one that contains segments closer to the initial hops. This overhead is due to the packet size and segment processing at each switch.

First we show in Figure 4.7 the average SLS using FSD strategy. It is possible to observe that networks organized as rings (including EDP topologies) require on average more segments to be added to the SL. This number tends to increase as the network grows, i.e., as more switches are added to this ring. By contrast, the BRITE and Fat Tree networks requires a small average SLS. It is interesting to notice that the average SLS of grids and Fat Trees decrease as the networks grow in number of switches. This is explained with the larger path redundancy that exists in BRITE networks and fat trees as more paths exists in the network, it is easier to find alternative paths.

Next, we study for each network topology which protocol step needs to be used to perform the recovery of every possible link failure: ECMP can reroute the traffic (Step 1); a LFA path is found (Step 2); one segment added to the SL is enough (Step 3); two segments are necessary (Step 4). The results of this analysis are presented in Figure 4.8 and Tables 4.1 and 4.2. The results show that ECMP is used rarely in most cases. This was more or less expected as in the majority of cases the weights assigned to links are random values, and therefore it is unlikely that two paths have exactly the same cost. It is possible to observe in Tables 4.1 and 4.2, that in BRITE topologies the values are closer to zero. The notable exception was as expected Fat Trees, which due to their regular structure and high redundancy, affords the use of ECMP in many links.

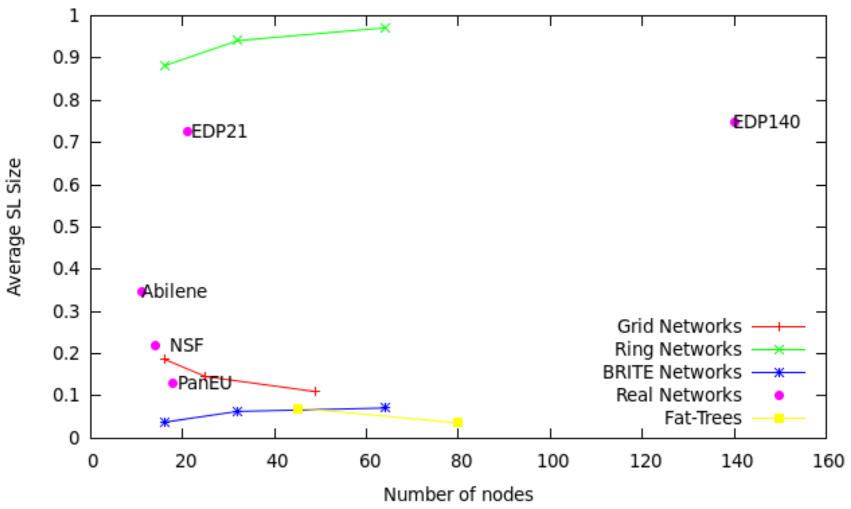


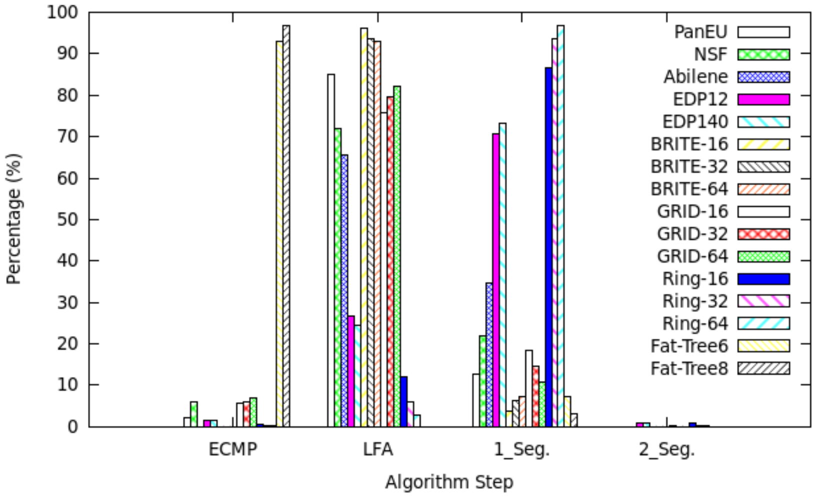
Figure 4.7: Average segment list size: SL size vs number of nodes using Fast Segment Drop.

Table 4.1: Algorithm Results Summary

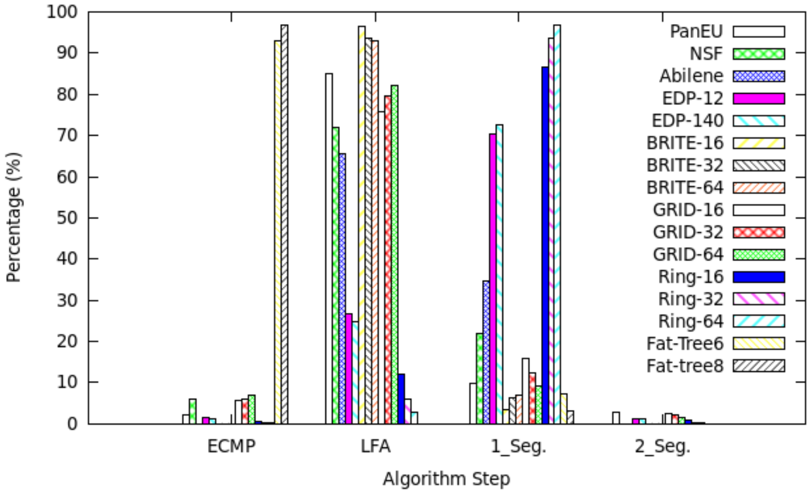
		SLS								
	N	Avg	Max	ECMP	LFA	1 Seg.	2 Segs.	Avg	Random	CAS
				(%)	(%)	(%)	(%)			
Pan EU	18	0.13	1	2.29	84.98	12.75	0.00	1.97	4.03	2.28
NSF	14	0.22	1	6.04	71.98	21.98	0.00	1.95	4.85	2.66
Abilene	11	0.35	1	0.00	65.45	34.55	0.00	2.29	3.18	2.47
EDP 21	21	0.72	2	1.35	27.12	70.69	0.82	2.99	5.78	3.49
EDP 140	140	0.75	2	1.34	24.53	73.00	0.84	3.24	5.74	3.60
BRITE	16	0.04	1	0.00	96.18	3.78	0.00	1.28	3.96	-
BRITE	32	0.06	2	0.00	93.62	6.32	0.00	1.62	5.73	-
BRITE	64	0.07	2	0.00	92.85	7.14	0.00	1.97	7.48	-
Grid	16	0.18	2	5.57	76.73	17.56	0.14	2.12	5.18	2.55
Grid	25	0.15	2	5.73	79.59	14.66	0.06	2.12	6.78	2.84
Grid	49	0.11	2	7.58	81.28	11.12	0.02	2.17	9.76	3.44
Ring	16	0.88	2	0.52	12.30	86.41	0.78	4.83	6.19	4.84
Ring	32	0.94	2	0.30	5.85	93.43	0.41	8.83	12.15	8.83
Ring	64	0.97	2	0.17	2.83	96.82	0.17	16.80	24.13	16.80
Fat Tree 6	45	0.07	1	92.88	0.00	7.12	0.00	2.00	4.67	2.15
Fat Tree 8	80	0.04	1	96.38	0.00	3.67	0.00	2.00	4.93	2.15

LFA, on the other hand, is very effective at supporting recoveries in most topologies, with the exception of ring based topologies (where EDP is also included). Due to the lack of redundancy, ring topologies require one more segment to be added because they are not well protected by LFAs alone. EDP topologies are multi-ring networks so they will inherit some problems of ring topologies. Thus, they require the use of 1-Segment more often, the values are above 70%. The graphs also give evidence that only in very few scenarios 2-Segments are necessary. The use of 2-Segments are very special corner cases which occur infrequently.

TI-LFA, however, causes the use of 2-Segment steps slightly more often due to always following the post-convergence path. This shows that using TI-LFA introduces a cost in terms of increasing overhead, by requiring more segments for recovery.



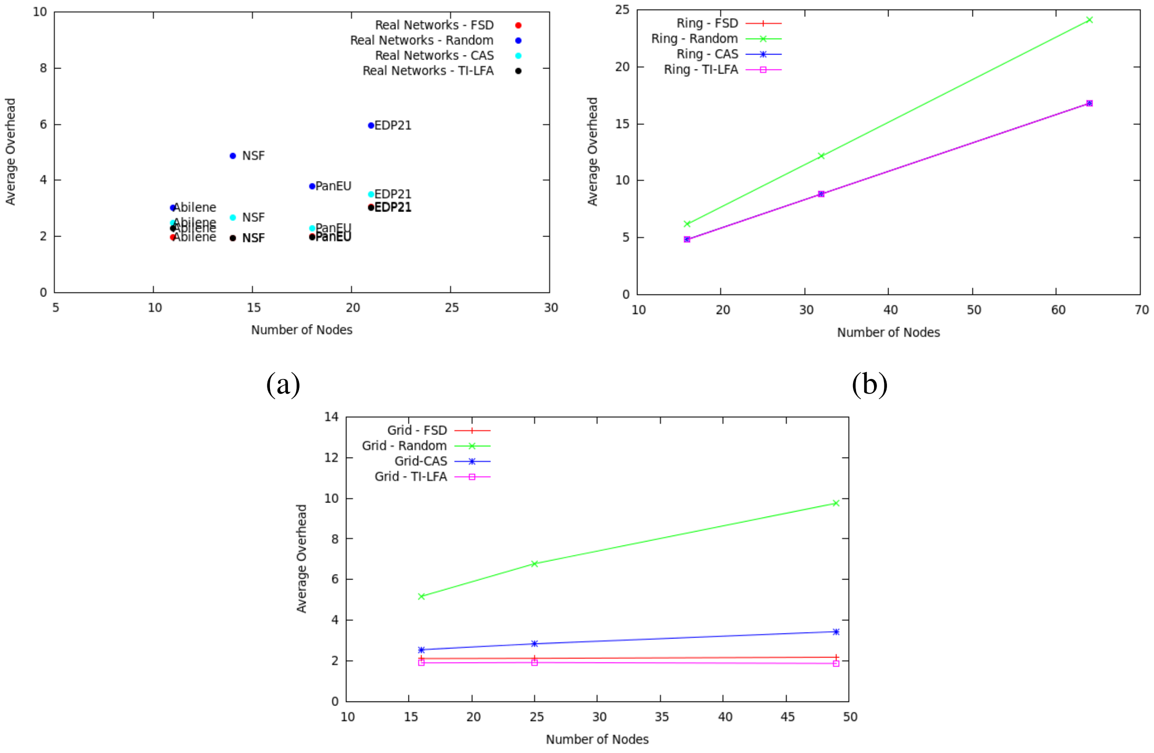
(a)



(b)

Figure 4.8: Percentage of steps used to find a backup path: In (a) it shows the percentage of times each step of the algorithm was used to find a backup path using the original algorithm and in (b) using TI-LFA

Interestingly, however, if we observe Figure 4.9, using more segments does not necessarily imply more overhead. This happens because while using FSD we do not force the packets to traverse the new shortest-path (usually the post-convergence path), therefore the packets carrying just one label may be routed through a longer path increasing the respective overhead. In Figure 4.9 (a) the values obtained for real network using TI-LFA and FSD are so close that is not possible to observe more than three values. The lowest value for NFS, PanEU and EDP-21 are overlapped. The same situation happens in 4.9 (c). Showing that the algorithms perform similarly. However in Ring topologies, the values for FSD, CAS, TI-LFA are the same. This happens because when a link fails there is only one viable path to arrive to the destination, causing the packet to travel the same path for all strategies. Using the CAS it will have an increasing overhead but at a cost of avoiding congested links.



(a)

(b)

(c)

Figure 4.9: Topologies Average Overhead: (a) real topologies, (b) Ring Topologies and (c) Grid topologies.

In Figure 4.10 and 4.11 we observe that with CAS we obtain a slight decrease in congestion when compared with TI-LFA for all topologies. As mentioned previously the values established for link capacity are 100Mbps. The majority of topologies show that 75% of values are below or slightly above 100Mbps. The only exceptions are EDP topologies that present more values above 100. This is once again due to the characteristics of the topology, rings do not present many alternative paths, therefore they have to follow a path that will become congested. This is the reason why we do not present results for rings since they are the same. As for grids we can observe that with TI-LFA the values keep increasing along with the topology size. in contrast, with CAS, the values decrease maintaining below 100Mbps.

Table 4.2: TI-LFA Algorithm Results Summary

	N	SLS Avg	 Max	ECMP (%)	LFA (%)	1 Seg. (%)	2 Segs. (%)	Avg. Ovh
Pan EU	18	0.16	1	2.29	84.97	9.80	2.94	1.97
NSF	14	0.22	1	6.04	71.98	21.98	0.00	1.95
Abilene	11	0.35	1	0.00	65.45	34.55	0.00	2.29
EDP 21	21	0.73	2	1.36	27.12	70.51	0.95	3.00
EDP 140	140	0.75	2	1.34	24.81	72.62	1.14	3.23
BRITE	16	0.03	2	0.01	96.56	3.35	0.00	1.25
BRITE	32	0.06	2	0.01	93.67	6.21	0.06	1.59
BRITE	64	0.07	2	0.00	93.02	6.92	0.05	1.94
Grid	16	0.20	2	5.57	76.73	15.14	2.56	1.90
Grid	25	0.17	2	5.73	79.55	12.56	2.17	1.92
Grid	49	0.13	2	7.58	81.29	9.55	1.59	1.88
Ring	16	0.88	2	0.52	12.30	86.41	0.78	4.83
Ring	32	0.94	2	0.30	5.85	93.43	0.41	8.83
Ring	64	0.97	2	0.17	2.83	96.82	0.17	16.80
Fat Tree 6	45	0.07	1	92.88	0.00	7.12	0.00	2.00
Fat Tree 8	80	0.04	1	96.33	0.00	3.67	0.00	2.00

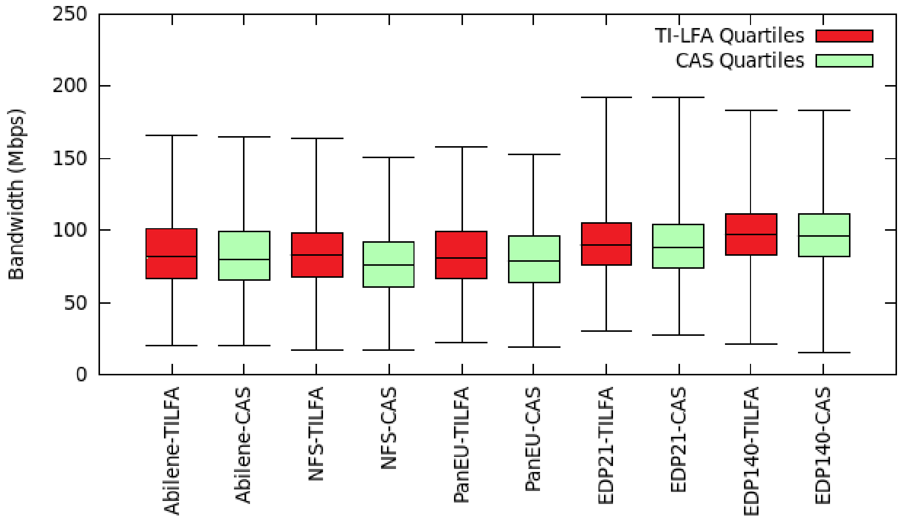


Figure 4.10: Bandwidth difference between TI-LFA and CAS in real topologies

4.4 Discussion

In this chapter we presented the evaluation to our strategies (FSD and CAS). In order to achieve this we had to implement a random selector and TI-LFA algorithm. We used three types of topologies: real topologies, random generated topologies and regular topologies.

The objectives of our evaluation were to analyze the average segment list size and

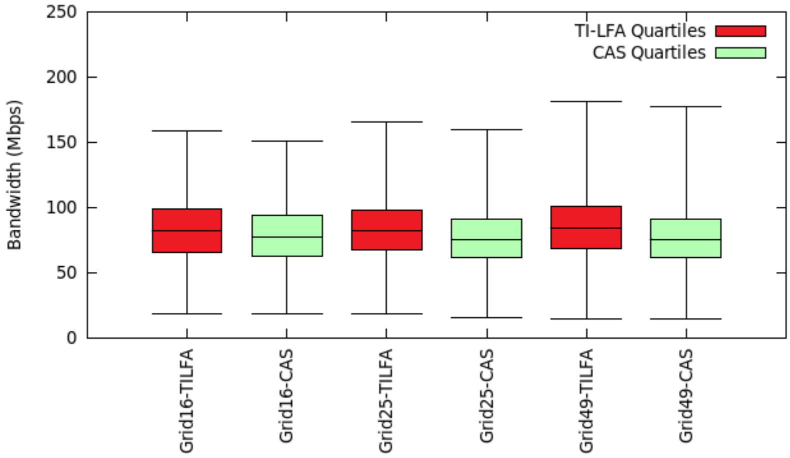


Figure 4.11: Bandwidth difference between TI-LFA and CAS in grid topologies

analyze the average packet overhead. We observed that most networks organized as rings require on average more segments to be added to the SL. However BRITE and Fat Tree networks requires a small average SLS due to larger path redundancy that they have.

We examined which protocol step needs to be used to perform the recovery of every possible link failure : The results show that ECMP is used rarely in most cases. LFA, on the other hand, is very effective at supporting recoveries in most topologies. Ring based topologies require the use of 1-Segment more often due to their characteristics. There is also evidence that in very few scenarios 2-Segments are necessary. While using TI-LFA, it causes the use of 2-Segment steps slightly more often due to always following the post-convergence path.

We can observe that with CAS we obtain a slight decrease in congestion when compared with TI-LFA for all topologies.

Chapter 5 - Conclusion

SG are a new generation of electric grids that are composed of smart meters, wired and wireless sensors and dispersed loads that need to be reliably connected through a network. A SG network requires a set of properties that range from high availability to security. Besides, they also need to be interoperable, able to connect and exchange data freely and transparently with many different types of devices. To provide such properties the communication architecture and data management play a fundamental role.

IP-based and MPLS communication technologies, often used in the core networks of smart grids, can achieve good levels of scalablility, security and be able to provide traffic engineering, but they present problems with due to the heterogeneous nature of the network elements or the complexity introduced by their supporting protocols (e.g., RSVP-TE).

SDN and SR are recent technologies that can contribute to solve the above problems. SDN decouples the data plane from the control plane, providing a centralized view of the network state. SR can support traffic engineering while maintaining a MPLS data plane, without requiring the use of complex protocols like LDP and RSVP-TE.

In this thesis, we proposed an algorithm that attempts to guarantee packet delivery after a single link failure. The implementation of our fast reroute algorithm resorts to SDN and SR. We also propose two novel strategies for segment selection: Fast Segment Drop that aims to minimize packet overhead and segment list size; and congestion Avoidance Segment, a strategy that provides traffic engineering by minimizing the maximum link load.

We compared our solutions with two other solutions (random and TI-LFA). Our proposal and TI-LFA share a similar general goal but the results show that our algorithm provides more flexibility when compared to TI-LFA as it allows for a wider segment choice. For instance, the results demonstrate that using CAS reduces the number of congested links when compared to TI-LFA. Using Fast Segment Drop it possible to achieve higher coverage (3% to 5%) with just one segment when compared to TI-LFA. Finally, these techniques can obtain 100% coverage after a single link failure with a small cost of packet overhead.

Glossary

AMI Advanced Metering Infrastructure.

BAN Building/Business Area Network.

DLFA Directed Loop Free Alternate.

DR Demand Response.

ECMP Equal-cost Multi-path.

FAN Field Area Network.

FIB Forwarding Information Base.

FIR Failure Insensitive Routing.

HAN Home Area Network.

HTML HyperText Markup Language.

IAN Industrial Area Network.

IED Intelligent electronic devices.

IETF Internet Engineering Task Force.

IGP Interior Gateway Protocol.

IP Internet Protocol.

IPFRR IP Fast Reroute.

IS-IS Intermediate System Intermediate System.

LDP Label Distribution Protocol.

LFA Loop Free Alternate.

LSP Label Switching Path.

LSR Label Switch Router.

MPLS Multiprotocol Label Switching.

MRC Multiple Routing Configurations.

NAN Neighbor Area Network.

OSPF Open Shortest Path First.

PLR Point of Local Repair.

PSN Partial Structural Network.

R3 Resilient Routing Reconfiguration.

RLFA Remote Loop Free Alternative.

RSVP-TE Resource Reservation Protocol with Traffic Engineering.

SDN Software Defined Networking.

SG Smart Grid.

SL Segment List.

SR Segment Routing.

UDP User Datagram Protocol.

WAN Wide Area Network.

Bibliography

[1] Jing Liu, Yang Xiao, Shuhui Li, Wei Liang, and Philip Chen. Cyber security and privacy issues in smart grids. *IEEE Communications Surveys & Tutorials*, 14(4):981–997, 2012.

[2] Diego Kreutz, Fernando Ramos, Paulo Verissimo, Christian Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[3] Scott Shenker, Martin Casado, Teemu Koponen, and Nick McKeown. The future of networking, and the past of protocols. *Open Networking Summit*, 20, 2011.

[4] Ian F Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. A roadmap for traffic engineering in sdn-openflow networks. *Computer Networks*, 71:1–30, 2014.

[5] Clarence Filsfils, Nagendra Nainar, Carlos Pignataro, Juan Cardona, and Pierre Francois. The segment routing architecture. In *Global Communications Conference*, pages 1–6. IEEE, 2015.

[6] Pierre Francois, Clarence Filsfils, Ahmed Bashandy, Bruno Decraene, and Stephane Litkowski. Topology independent fast reroute using segment routing. 2016. https://tools.ietf.org/id/draft-francois-rtgwg-segment-routing-ti-lfa-01.txt.

[7] Vijay K Sood, Daniel Fischer, JM Eklund, and Tim Brown. Developing a communication infrastructure for the smart grid. In *Proceedings of the Electrical Power & Energy Conference (EPEC)*, pages 1–7. IEEE, 2009.

[8] Tong Xiaoyang, Liao Guodong, Wang Xiaoru, and Zhong Shan. The analysis of communication architecture and control mode of wide area power systems control. In *Proceedings Autonomous Decentralized Systems*, pages 59–65. IEEE, 2005.

[9] Jon Postel. Internet protocol. STD 5, RFC Editor, September 1981. `http://www.rfc-editor.org/rfc/rfc791.txt`.

[10] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, RFC Editor, January 2001. `http://www.rfc-editor.org/rfc/rfc3031.txt`.

[11] Fernando Ramos, Diego Kreutz, and Paulo Verissimo. Software-defined networks: On the road to the softwarization of networking. *Cutter IT journal*, 2015.

[12] Thilo Sauter and Maksim Lobashov. End-to-end communication architecture for smart grids. *IEEE Transactions on Industrial Electronics*, 58(4):1218–1228, 2011.

[13] Hassan Farhangi. The path of the smart grid. *IEEE power and energy magazine*, 8(1):18–28, 2010.

[14] Jon Postel. Transmission control protocol. STD 7, RFC Editor, September 1981. `http://www.rfc-editor.org/rfc/rfc793.txt`.

[15] J. Postel. User datagram protocol. STD 6, RFC Editor, August 1980. `http://www.rfc-editor.org/rfc/rfc768.txt`.

[16] James F Kurose and Keith W Ross. *Computer networking: a top-down approach*. Pearson, 2013.

[17] Charles Hedrick. Routing information protocol. RFC 1058, RFC Editor, June 1988. `http://www.rfc-editor.org/rfc/rfc1058.txt`.

[18] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. In *Proceedings of INFOCOM 2000. Nineteenth annual joint conference of the IEEE computer and communications societies*, volume 2, pages 519–528. IEEE, 2000.

[19] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional ip routing protocols. *IEEE Communications Magazine*, 40(10):118–124, 2002.

[20] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[21] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Stuart.Stephen, and Amin Vahdat. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013.

[22] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 15–26. ACM, 2013.

[23] Nanxi Kang, Monia Ghobadi, John Reumann, Alexander Shraer, and Jennifer Rexford. Efficient traffic splitting on sdn switches. In *Proceedings of CoNEXT '15 the 11th ACM Conference on Emerging Networking Experiments and Technologies, Article No. 6*, 2015.

[24] Randeep Bhatia, Fang Hao, Murali Kodialam, and T.V. Lakshman. Optimized network traffic engineering using segment routing. In *Proceedings of the Conference on Computer Communications (INFOCOM)*, pages 657–665. IEEE, 2015.

[25] Luca Davoli, Luca Veltri, Pier Luigi Ventre, Giuseppe Siracusano, and Stefano Salsano. Traffic engineering with segment routing: Sdn-based architectural design and open source implementation. In *Proceedings of the Fourth European Workshop on Software Defined Networks*, pages 111–112. IEEE, 2015.

[26] Alessio Giorgetti, Andrea Sgambelluri, Francesco Paolucci, and Piero Castoldi. Reliable segment routing. In *Reliable Networks Design and Modeling (RNDM), 2015 7th International Workshop on*, pages 181–185. IEEE, 2015.

[27] Alessio Giorgetti, Andrea Sgambelluri, Francesco Paolucci, Filippo Cugini, and Piero Castoldi. Demonstration of dynamic restoration in segment routing multilayer sdn networks. In *Proceedings of the Optical Fiber Communication Conference*, pages Th4G–4. Optical Society of America, 2016.

[28] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. Network architecture for joint failure recovery and traffic engineering. *ACM SIGMETRICS Performance Evaluation Review*, 39(1):97–108, 2011.

[29] Amund Kvalbein, Audun Fosselie Hansen, Tarik Čičic, Stein Gjessing, and Olav Lysne. Multiple routing configurations for fast ip network recovery. *IEEE/ACM Transactions on Networking (TON)*, 17(2):473–486, 2009.

[30] Sanghwan Lee, Yinzhe Yu, Srihari Nelakuditi, Zhi-Li Zhang, and Chen-Nee Chuah. Proactive vs reactive approaches to failure resilient routing. In *Proceedings of the Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1. IEEE, 2004.

[31] Baohua Yang, Junda Liu, Scott Shenker, Jun Li, and Kai Zheng. Keep forwarding: Towards k-link failure resilient routing. In *Proceedings of the Conference on Computer Communications (INFOCOM)*, pages 1617–1625. IEEE, 2014.

[32] Minlan Yu, Jennifer Rexford, Michael J Freedman, and Jia Wang. Scalable flow-based networking with DIFANE. *ACM SIGCOMM Computer Communication Review*, 40(4):351–362, 2010.

[33] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. R3: resilient routing reconfiguration. In *Proceedings of SIGCOMM Computer Communication Review*, volume 40, pages 291–302. ACM, 2010.

[34] Mike Shand and Stewart Bryant. Ip fast reroute framework. RFC 5714, RFC Editor, January 2010. http://www.rfc-editor.org/rfc/rfc5714.txt.

[35] Alia Atlas and Alex Zinin. Basic specification for ip fast reroute: Loop-free alternates. RFC 5286, RFC Editor, September 2008. https://tools.ietf.org/rfc/rfc5286.txt.

[36] Ping Pan, George Swallow, and Alia Atlas. Fast reroute extensions to rsvp-te for lsp tunnels. RFC 4090, RFC Editor, May 2005. https://tools.ietf.org/html/rfc4090.

[37] Stephane Litkowski, Bruno Decraene, Clarence Filsfils, and Kamran Raza. Interactions between lfa and rsvp-te. Internet-Draft draft-litkowski-rtgwg-lfa-rsvpte-cooperation-01, February 2013. http://www.ietf.org/internet-drafts/draft-litkowski-rtgwg-lfa-rsvpte-cooperation-01.txt.

[38] C. Hopps. Analysis of an equal-cost multi-path algorithm. RFC 2992, RFC Editor, November 2000. http://www.ietf.org/mail-archive/web-old/ietf-announce-old/current/msg09968.html.

[39] Ina Minei and Julian Lucek. *MPLS-enabled applications: emerging developments and new technologies*. John Wiley & Sons, 2010.

[40] Clarence Filsfils, Stefano Previdi, Bruno Decraene, Stephane Litkowski, and rjs@rob.sh. Segment routing architecture. Internet-Draft draft-ietf-spring-segment-routing-04, IETF Secretariat, July 2015. http://www.ietf.org/internet-drafts/draft-ietf-spring-segment-routing-04.txt.

[41] Stewart Bryant, Clarence Filsfils, Stefano Previdi, Mike Shand, and Ning So. Remote loop-free alternate (lfa) fast re-route (frr). Internet-Draft draft-ietf-rtgwg-remote-lfa-11, IETF Secretariat, January 2015. http://www.ietf.org/internet-drafts/draft-ietf-rtgwg-remote-lfa-11.txt.

[42] Anica Bukva, Ramon Casellas, Ricardo Martínez, and Raül Muñoz. A dynamic path-computation algorithm for a gmpls-enabled multi-layer network. *Journal of Optical Communications and Networking*, 4(6):436–448, 2012.

[43] Alessio Giorgetti, Piero Castoldi, Filippo Cugini, Jeroen Nijhof, Francesco Lazzeri, and Gianmarco Bruno. Path encoding in segment routing. In *Proceedings Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.