Pós-Graduação em Ciência da Computação

WALBER JOSÉ ADRIANO SILVA

An Architecture to Evolve the Inter-Domain Routing Using Software-Defined Networking

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

WALBER JOSÉ ADRIANO SILVA

An Architecture to Evolve the Inter-Domain Routing Using Software-Defined Networking

Tese apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciências da Computação.

Área de Concentração: Redes de Computadores.

Orientador: Prof. Dr. Djamel Fawzi Hadj Sadok

Recife

2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586a Silva, Walber José Adriano
An architecture to envolve the inter-domain routing using software-defined networking / Walber José Adriano Silva. – 2019.
144 f.: il., fig., tab.

Orientador: Djamel Fawzi Hadj Sadok.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências.

1. Redes de computadores. 2. SDN. 3. BGP. I. Sadok, Djamel Fawzi Hadj (orientador). II. Título.

004.6 CDD (23. ed.) UFPE- MEI 2019-088

Walber José Adriano Silva

"An Architecture to Evolve the Inter-Domain Routing Using Software-Defined Networking"

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 15/03/2019.

____

Orientador: Prof. Dr. Djamel Fawzi Had Sadok

BANCA EXAMINADORA

____

Prof. Dr. Paulo Romero Martins Maciel
Centro de Informática /UFPE

____

Prof. Dr. Arthur de Castro Callado
Departamento de Computação / UFC - Campus Quixadá

____

Prof. Dr. Eduardo Luzeiro Feitosa
Instituto de Computação / UFAM

____

Prof. Dr. Reinaldo Cézar de Morais Gomes
Departamento de Sistemas e Computação / UFCG

____

Prof. Dr. Rafael Roque Aschoff
Instituto Federal de Pernambuco / Campus Palmares

I dedicate this thesis to all my family, friends and professors who gave me the necessary support to get here.

ACKNOWLEDGEMENTS

First of all, to God for the innumerable bestowed gifts provided that enable me to accomplish and complete this work.

This thesis is the result of a long journey, with the support of my family, my friends, and my beloved, Dayane. As it could not be different, I dedicate to them this work, and I thank all the affection, understanding, and incentive.

To Dayane, for always being (literally) by my side. She has been my source of inspiration and the greatest encouragement for surpassing the hard moments. Believe me, I would not have the strength and motivation I have without your fellowship, dedication, and love. To her, I will **always** be grateful.

To my family, for understanding and always encouraging my decision to do the doctorate, recognizing the importance of it for my studies and supporting any decision that was taken by me. I thank my parents, Leandro and Socorro, for believing that this was the right path to follow. For the love, affection, and words of encouragement that allows me to fly higher and higher in pursuit of my goals. I thank my best friend and brother Thiago, for the support, motivation, time, and his unconditional love.

To professors Gustavo Motta (UFPB, Brazil) and Nuno Preguiça (NOVA, Portugal) for the words of encouragement that motivated me to join the doctorate course.

To my colleagues at the Informatics Center - CIn and the Nucleus of Information Technology - NTI who helped in the development of research activities, especially Alexandre (whom I wish to succeed in all his endeavors, too) and Edivaldo.

To all the teachers of the CIn-UFPE, for providing the knowledge, not only rational, but the manifestation of the character and affectivity of education in the process of professional training. Especially the teachers Djamel, Judith, Paulo Maciel and Kelvin for the knowledge transmitted in the disciplines of doctorate.

To the government employees of UFPE and UFRN who have always been helpful, quick, and zealous in meeting the demands of my doctorate. I would like to highlight the Secretary Office of CIn-UFPE, the CCEN-UFPE Library and Central-UFPE Library, and the Network and Support Office - CERES-UFRN, the Manage Office of CERES-UFRN, PRH-UFRN, and the CERES-UFRN Library.

To the whole GPRT staff for socializing, advice, and assistance during the doctoral-attendance meetings. In particular, Professor Patricia Endo for the exquisite corrections and comments in my thesis proposal, and also to GPRT secretaries Ana Carolina and Roxana Calheiros for being very helpful and attentive to me.

Especially my advisor Professor Djamel Sadok, for the opportunity given, for the exchange of ideas, for the teachings, for the critical vision of my work, being always willing to discuss the progress of the research activities and providing guidelines, even with all their attributions.

I also would like to thank Professors Paulo Maciel, Arthur Callado, Eduardo Feitosa, Reinaldo Gomes, and Rafael Aschoff, as well as the anonymous reviewers from multiple conferences and periodic (ICOIN-2017, HPSR-2017, SBrT-2017, ICNC-2018, JONS-2018, and Information-2018) for their valuable comments and suggestions to improve my academic works.

And if you were not mentioned here, but collaborated directly or indirectly in my doctorate, I sincerely thank you. This is a life stage that ends so that another begins because a good life is one in which it is worth living it, and I enjoyed every little moment of this journey (from the bad to the good times). Of course, I will take what I have learned with each of you for the rest of my life.

“Seja você quem for, seja qual for a posição social que você tenha na vida, a mais alta ou a mais baixa, tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá. De alguma maneira você chega lá.” (Silva, Ayrton Senna da, [199-?])

ABSTRACT

The Border Gateway Protocol (BGP) is the current state-of-the-art protocol for inter-domain routing between Autonomous System (AS). The BGP follows the traditional network paradigm, where the network logic is embedded into the network hardware, new network capabilities are defined only in the protocol design phase, solutions have a distributed nature, and network management is often manually executed. This approach makes challenging to evolve inter-domain routing because architecture becomes very dependent on the protocol, and consequently adding new features are inherently hard to introduce into the network. With Software-Defined Networking (SDN) paradigm, new architectures can emerge to overcome inter-domain routing limitations (e.g., limited mechanisms to control inbound traffic of a given AS). Thus, this work offers a new architecture to manage traffic in the inter-domain using SDN technologies. The architecture explores direct inter-domain communication to exchange control information as well as the functionalities of the OpenFlow protocol. It also addresses the challenge of adopting SDN technologies in the inter-domain environment, such as resilience mechanisms for link failures, management of flows and avoids inconsistency in SDN stateful application. Furthermore, after the depiction of the architecture, different scenarios are proposed for evaluation of the architecture capabilities. The results indicate the potential of the architecture to evolve the inter-domain routing, once it provides new mechanisms to perform Traffic Engineering (TE) and addresses solutions to mitigate the limitations of using SDN technologies in the inter-domain environment.

Keywords: SDN. BGP. OpenFlow.

RESUMO

O Border Gateway Protocol (BGP) é o principal protocolo para roteamento entre Sistemas Autônomos (AS). O BGP segue o paradigma de rede tradicional, onde a lógica da rede é embarcada no hardware, os novos recursos de rede são definidos apenas na fase de projeto do protocolo, as soluções de rede têm natureza distribuída, o gerenciamento da rede é executado manualmente, entre outras características. Essa abordagem de projetar redes dificulta o desenvolvimento de novas lógicas de roteamento entre domínios, pois a arquitetura se torna muito dependente do protocolo e, consequentemente, a adição de novos recursos é inerentemente difícil de ser introduzida na rede. Com o paradigma das Redes Definidas por Software (SDN), novas arquiteturas podem surgir para superar as limitações de roteamento entre domínios. Assim, esta tese oferece uma arquitetura para gerenciar o tráfego de rede entre domínios com o uso de tecnologias SDN. A arquitetura explora a comunicação direta entre domínios para trocar informações de controle, bem como as funcionalidades providas pelo protocolo OpenFlow. Ademais, também é abordado o desafio em adotar tecnologias SDN no ambiente inter-domínio, como: resiliência da conectividade entre domínios no caso da falha de enlaces; o gerenciamento de fluxos em redes SDN; e, a mitigação de inconsistência de estado em aplicações SDN. Desta forma, após a descrição da arquitetura, diferentes cenários foram propostos e avaliados. Os resultados indicam o potencial da arquitetura em evoluir o roteamento entre domínios, uma vez que fornece novos mecanismos para executar tarefas de engenharia de tráfego e aborda soluções para atenuar as limitações do uso de tecnologias SDN no ambiente inter-domínio.

Palavras-chaves: SDN. BGP. OpenFlow.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1 INTRODUCTION

The last decade has been marked by profound advances in Information and Communication Technologies (ICT). Thereby, it is expected that computer networks will continue to play an important role in the adoption of new ICT, for example, Internet of Things (IoT), 5G, Software-Defined Networking (SDN) and others (HUANG et al., 2017). Thus, facing the great diversity of technologies, a growing problem is in the inter-domain.

The inter-domain network environment is a collection of tens of thousands of independently operated networks. In the Internet jargon, these networks are called Autonomous System (AS)s. An AS can be an Internet Service Provider (ISP), a Campus, a Content Provider, or any other independently operated network that can establish its routing policies and is perfectly capable of define how the network should be operated. Besides, ASs can connect with each other as they execute an inter-domain routing protocol to exchange network reachability information. And the Border Gateway Protocol (BGP) (REKHTER; LI; HARES, 2006) is current standard for inter-domain routing on the Internet and provides reachability information among the ASs.

The BGP is the "glue" that allows different ASs to reach each other networks. An AS does tune its BGP configurations to express policies that reflect how the AS connects to others for accommodating its business requirements. Besides that, this protocol has been allowing the Internet exchange reachability information and network traffic since its early stages.

Furthermore, the Internet topology and traffic patterns have been changing since its inception, from a pure hierarchical topology (well-defined tree structure) to a flat AS-level topology (more connected and without a well defined structured topology) (LUCKIE et al., 2013). One cause of the Internet topology changing is the expansion of Content Delivery Network (CDN) and Internet Exchange Point (IXP). They have been increasing the path diversity of the Internet, where CDN and IXP introduced additional AS connectivity to intensify Internet traffic among ASs (LABOVITZ et al., 2010; AGER et al., 2012).

1.1 MOTIVATION

BGP is one of the most successful protocols on the Internet and it has a great capacity to scale but it is nonetheless one of the most inflexible and "ossified" protocols (KOTRONIS, 2015; HAKIRI et al., 2014; CHOWDHURY; BOUTABA, 2010). Many researchers have indicated issues related to BGP since its inception. A few drawbacks of BGP are the lack of end-to-end guarantees of Quality of Service (QoS) (YANNUZZI; MASIP-BRUIN; BONAVENTURE, 2005), complex network management (FEAMSTER et al., 2005), long convergence time (YANNUZZI; MASIP-BRUIN; BONAVENTURE, 2005), security issues (KEVIN et al., 2010), and limitations of routing policy enforcement (GIOTSAS; LUCKIE; HUFFAKER, 2014). For example, BGP does not have robust Traffic Engineering (TE) techniques for control inbound traffic (SILVA; SADOK, 2017) and the current Internet routing table explosion is a side effect of the application of BGP traffic engineering tasks by multi-homed ASs, in which ASs desiring to increase the reliability of their domains by advertising multiple subnets of their network prefixes through different AS neighbors (POTAROO.NET, 2016).

The Internet is intrinsically heterogeneous and distributed, where network operators execute independent network management task into their AS (LIN et al., 2014). And the lack of active BGP mechanisms to apply traffic engineering on the traffic contributes to the over-provisioned ASs infrastructure, leading to an inter-domain environment consisting of expensive, high-performance and specialized hardware. Consequently, those network resources are around 30% to 60% (HONG et al., 2013) underutilized to cope with disruption or unpredictable changes of connectivity in the inter-domain environment.

One way to fully utilize the network resources between different domains will be exploring the existed multi-paths of the Internet. However, the BGP's destination-based forwarding paradigm limits the granularity of distributing network traffic among the multiple paths of the current Internet topology. Hence, BGP is not capable of exploring the full potential of path diversity in the current Internet, because it just computes one "best" next hop for each network prefix. New technologies in use such as 5G (GUPTA; JHA, 2015) and the IoT (AL-FUQAHA et al., 2015) have very ambitious requirements (for example, low latency and high bandwidth links), and that will stress the network's capabilities, especially in the inter-domain environment. Addressing those network requirements within an Internet not designed to support them is a challenging task. Therefore, new network architectures and technologies can emerge to explore the full potential of the current Internet infrastructure (SINGH; DAS; JUKAN, 2015).

The difficultly to evolve inter-domain routing is caused by the process of "ossification", where the architecture becomes very dependent on the protocol, and new features are inherently hard to introduce into the network. Specifically for the BGP, the "ossification" is due to economic reasons combined with the fact that backward compatibility has to be assured since there is no flag day to switch to a new architecture. Thus, all BGP appliances have to execute the same version of the protocol to operate appropriately, or anomalies in the network may occur (e.g., BGP black holes). Hence, a new feature for the BGP protocol has to have a minimum integration within heterogeneous networks and interoperate through different administrative domains (e.g., the Internet). These requirements have frustrated new proposals to evolve BGP and the ecosystem of inter-domain routing (SILVA; SADOK, 2017).

Although BGP is a scalable protocol and widely deployed, it does not explore the full potential of the Internet once it is a single path protocol with unclear TE mechanisms, where the collaboration between domains (especially multiple AS-paths away) is a difficult

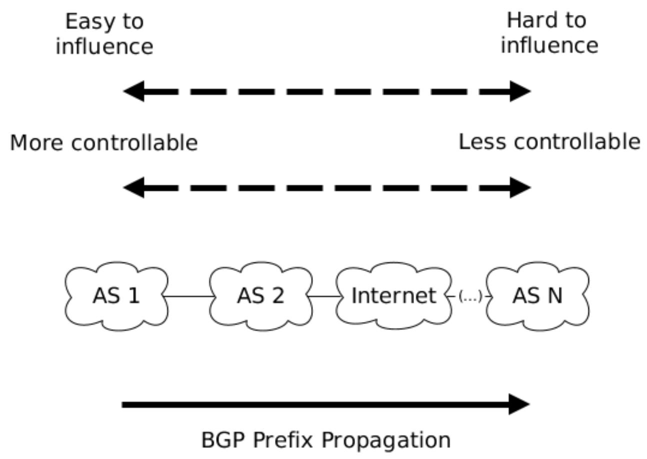


Figure 1 – Current BGP behavior, where as far as a prefix advertised by an AS propagates on the Internet, it became less controllable (how traffic should be handled) and harder to influence by the AS that originate it.

task (CARDONA et al., 2016). Thus, Figure 1 depicts the motivation of this Thesis, where a BGP prefix from a AS propagates into the Internet, it becomes less controllable, and harder is the effect of BGP TE techniques. **The vision for an Internet where the AS has more control and influence to its prefixes advertised on the Internet is the direction of this Thesis and the path to reach this goal is to work on the current architecture of the inter-domain environment**. This vision is depicted in Figure 2.

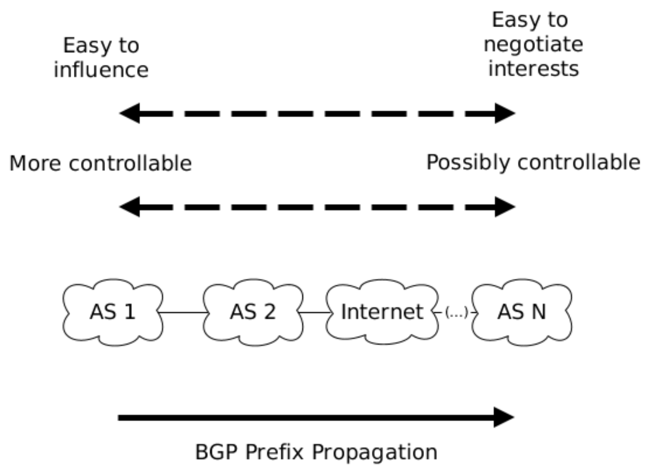


Figure 2 – Vision of this Thesis and what the proposed architecture in this Thesis is trying to achieve for the Internet.

One promising approach to enable network architectures to evolve is SDN. SDN emerged as a paradigm that allows network engineers to elaborate new network designs and includes desirable features for this evolution such as open programmable interfaces at networking devices, the separation between control and forwarding planes, and innovation is in applications executed inside the control planes (MENDIOLA et al., 2017; NUNES et al., 2014; KREUTZ et al., 2015; SILVA, 2018b).

A notable SDN technology is the OpenFlow (OF) protocol. OF is an open programmable interface for SDN that gained momentum in academia and industry over the last decade (MCKEOWN et al., 2008). OF is flow-oriented, meaning that the sequence of packets identified by a set of common header fields follows a given path. Thus, OF can provide fine-grain forward capabilities to the network by matching multiple fields of Transmission Control Protocol and Internet Protocol (TCP/IP).

However, the benefits of applying SDN technologies in the inter-domain do not come for free. Creating an efficient quantity of flows inside an OF network is a challenging management task due to the limitation of the resources (e.g. switches memory) and the signaling overhead between data and control plane elements (OpenFlow switches and controllers, respectively) (NGUYEN et al., 2016).

Therefore and summarizing, the following research questions have been investigated in this Thesis:

- **What are the requirements for a new architecture for inter-domain routing?** One major component of the inter-domain routing system is the control plane. Then, through a survey that gives an original classification of previous proposals to evolve the inter-domain routing control plane are discussed the challenges to the evolution of current and new control planes.

- **How to apply traffic engineering tasks between different SDN domains?** BGP has some mechanisms to apply traffic tasks, for example, Multi-Exit Discriminator (MED), Communities, AS-Path Prepending and others. However, the effectiveness of applying these mechanisms is often unclear and is also not guaranteed since they are indirect approaches. This Thesis proposes and investigates a new architecture and mechanisms to apply traffic engineering tasks between domains.

- **How to manage flows inside an SDN domain?** Different from traditional networks, in which TE tasks uses an Interior Gateway Protocol (IGP), e.g. Open Shortest Path First (OSPF) for manage routing information inside the domain, the SDN network relies on proper management of flows. Management of network flows in SDN is a top concern because of restrictions in network device memories (such as size, energy consumption and so forth) and the signaling overhead to control the network. For OpenFlow networks, it is an issue to decide where and when a network flow rule has to be installed (SILVA, 2018c).

1.2 OBJECTIVES

Considering the motivations described, the primary objective of this Thesis is to propose, elaborate, design, implement and evaluate an architecture to evolve the inter-domain routing using SDN to enable new TE tools and mitigate the weight of adopting SDN technologies for being used in the inter-domain environment. Furthermore, the architecture is backward compatible with BGP and composed by SDN technologies that allow the coordination of ASs to perform TE tasks. Some scenarios are presented and analyzed through simulation and emulation techniques to validate the architecture.

The specific goals of this Doctoral Thesis are:

- Classify previous works in the inter-domain routing;
- Evaluate and discuss an architecture to evolve the inter-domain routing using SDN;
- Depict mechanisms for properly manage flows in SDN networks, taking into account resilience requirements, keeping the trade-off between the number of rules installed and signaling between a SDN controller and switches, and avoiding inconsistency in network state of SDN applications.

1.3 STATEMENT OF THE CONTRIBUTIONS

The main contributions of this thesis are:

- Providing an original classification of previous proposals to evolve the control plane of the inter-domain routing. And also, discussing about the challenges to the evolution of current and new control planes (Chapter 3).
- Presenting and describing an architecture that uses OF networks and is compatible with current inter-domain routing protocol BGP (see Chapter 4);
- Presenting different schemes to create and install flows (reactive, proactive and active approaches), avoiding inconsistency in OpenFlow stateful applications, as well as a resilience scheme to mitigate link failures in OpenFlow networks (Chapter 4);
- Exploring OF features for network traffic engineering tasks in inter-domain routing (Chapter 5);
- Analyzing how the traffic can be managed using the proposed architecture and BGP (Chapter 5);

Therefore, the contribution for the computer network community is not limited by the text of this Thesis itself, but also expands to the published papers. See the Reference chapter to read the papers published as a result of this Doctoral Thesis.

1.4 OUTLINE OF THESIS

The remaining structure of this Thesis is: Chapter 2 provides the basic knowledge and concepts about SDN and inter-domain routing using BGP. Then, Chapter 3 gives an original classification of previous proposals to evolve the control plane of the inter-domain routing and also discusses the challenges to the evolution of current and new control planes. Afterward, Chapter 4 presents the architecture to manage inter-domain routing and mechanisms that address the difficulties of managing rules inside SDN networks. Then, Chapter 5 evaluates the proposed architecture and Chapter 6 presents the limitations, future works and final thoughts of this work.

2 FUNDAMENTALS

This chapter presents some technical background and an overview of Software-Defined Networking (SDN), the OpenFlow protocol (an SDN technology) and the Border Gateway Protocol (BGP). The main idea is to introduce the fundamentals concepts that are used in the remaining of the work. If the reader is already comfortable with those fundamentals, it can skip this chapter without loss of understanding about this Thesis.

2.1 OVERVIEW OF SOFTWARE-DEFINED NETWORKING

Traditional network protocols have been developed and deployed by combining software and hardware into network devices (*e.g.* TCP/IP). Indeed that approach makes network management a complex and laborious task, once to implement the desired high-level administration network policies it is required to configure each network device and perform low-level instructions/commands (often vendor-specific). Depending on the number of network devices, making such procedures are slow, challenging, and error-prone (KREUTZ et al., 2015).

Software-defined networking (SDN) is emerging as a network paradigm which proposes a separation of software-hardware from devices (vertical integration), generalizes network devices and functions, enables programmability to network, centralizes network management task (NUNES et al., 2014). A major feature of SDN is the decoupled control and data plane. The centralization logic of SDN is aware of network state; can enforce network policies, routing decisions, forwarding information, and so forth. Figure 3 presents the SDN architecture divided in three layes: Application Layer, Control Layer and Infrastructure Layer.

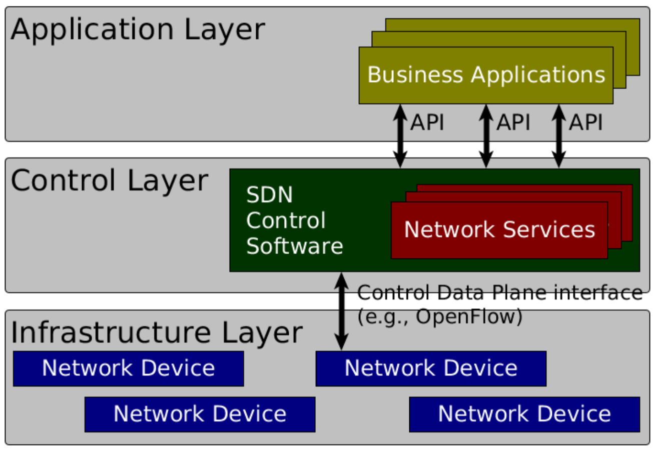


Figure 3 – SDN Architecture.

Due to the logical centralization of the control of an SDN network, availability has become an attribute of prime concern. Because the network programmability is done through software applications on the control plane, not more on underlying data plane devices (KREUTZ et al., 2015), data plane elements must receive instructions from the control planes elements when required.

2.2 OVERVIEW OF OPENFLOW PROTOCOL

There are some instances of the concepts of SDN (FEAMSTER; REXFORD; ZEGURA, 2014), and one of them is the OpenFlow protocol (MCKEOWN et al., 2008). The OpenFlow protocol (a standardized open interface) allows an OpenFlow controller to program the Forwarding Information Base (FIB) of OpenFlow switches (MCKEOWN et al., 2008). An OpenFlow network uses the concept of flow to carry traffic inside the network. Flow is a sequence of packets sent from a particular source to a particular destination following a given path (KREUTZ et al., 2015), where the packets have the same fields values. And a path is an ordered sequence of OpenFlow Switches and links from the origin to the destiny. In an OpenFlow Switch, the flow entry match fields, in a flow table, include layer 2, 3 or 4 header information, the ingress port, or the metadata values (PFAFF et al., 2012). The information about how OpenFlow Switches should behave comes from the OpenFlow controller.

Hence, to better understand the terminology and behavior of the elements on OpenFlow specification, this Section describes a brief overview of the protocol. If the reader requires more details about OpenFlow, we encourage to read the public specification of the protocol (OPENNETWORKING, 2014).

2.2.1 OpenFlow Switch Components

The main components of an OpenFlow architecture is described at Figure 4. Through multiple connections, an OpenFlow Switch (data plane element) can handle requests from the network by consulting one or more OpenFlow Controller. Based on the logic inside the controller, the behavior of an OpenFlow network is defined. An OpenFlow Switch has multiple internal elements to treat flows. A brief description of those components is (OPENNETWORKING, 2014):

- Port: where packets enter and exit on the pipeline;
- Flow Table: is the intern structure on the OpenFlow Switch that contain flows entries. Actions and matches for flows are storage at it. Furthermore, a flow table is what compose a pipeline;
- Pipeline: the set of linked flow tables. The main goal of a pipeline is to provide matching, forwarding, and packet modification on flows inside the OpenFlow Switch;

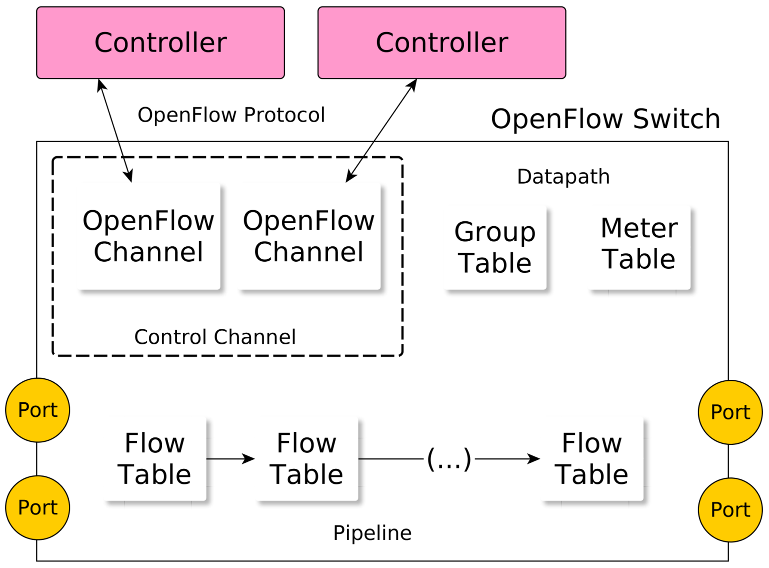


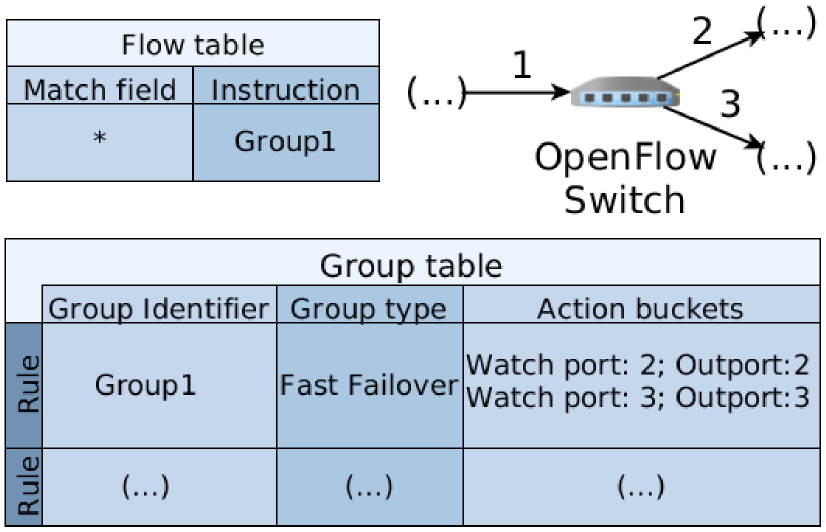
Figure 4 – Components of an OpenFlow Switch.

- Group Table: is where a list of action buckets and some means of choosing one or more of those buckets to apply on a per-packet approach;
- Meter Table: contains information about measurements of the packets that travels flow tables;
- Controller: an entity that send control information to an OpenFlow Switch;
- OpenFlow Channel: the interface for communication and management between an OpenFlow Switch and an OpenFlow Controller.

The OpenFlow protocol provides features and abstractions to enable a controller to program the OpenFlow Switch. Additionally, the current version of OpenFlow allows some management tasks to be done inside the OpenFlow Switch without requiring the controller intervention. The Group Table is a feature that enables the OpenFlow Switches to add new features. For example, OpenFlow Fast Failover Group Table (FF) enables detection and actions to the mitigation of link failures (LIU et al., 2014a).

2.2.2 OpenFlow Fast Failover Group Table

As explained in the work of (CASCONE et al., 2016), which proposed an SDN resilience mechanisms implemented in hardware that use OpenState (BIANCHI et al., 2014) (an extension of OpenFlow specification for stateful packet processing), the resilience mechanism deployed into hardware have notorious minimal unavailability time in comparative with software approaches. Thus, since the OpenFlow version 1.1, the OpenFlow specification provides the implementation of Fast Failover Group Table (FF) (PFAFF et al., 2011), which is a resilience mechanism deployed into an OpenFlow switch which could change primary paths to backup paths in the presence of link failures. That local action occurs without the switch requiring to communicate to the OpenFlow controller. The main goal of FF is to decrease the restoration time after a failure and avoid the overhead communication between switch and controller.



Flow table	
Match field	Instruction
*	Group1

	Group table		
	Group Identifier	Group type	Action buckets
Rule	Group1	Fast Failover	Watch port: 2; Outport:2 Watch port: 3; Outport:3
Rule	(...)	(...)	(...)

Figure 5 – Scheme of the Fast Failover.

Figure 5 depicts an scheme for using FF. Suppose an OpenFlow switch has the flow table as indicated, in which the instruction of flow is to apply the FF group type. Thus, the switch will autonomously monitor the first port in the *action buckets* seeking for a change in the first watch port. If a link failure occurs with that network port interface, the status of the port will not have the bit OFPPS_LIVE enable, and the switch will apply the next action in that *action buckets*. In the example of Figure 5, if port 2 goes down (caused by a failure), the switch stops forwarding traffic to port 2 and stars to forwarding traffic to port 3. Whenever a link failure occurs, the controller must gain awareness about the underly topology.

2.3 FLOW CREATION

Different from traditional networks, in which Traffic Engineering (TE) tasks uses Interior Gateway Protocol (IGP) (e.g. Open Shortest Path First (OSPF)) for managing traffic, the SDN networks relies on properly management of flows. Management of network flows in SDN is a top concern because of restrictions in network devices' memory (such as size, energy consumption, and so forth) and the signaling overhead to control the network.

In an OpenFlow network, the flows are deployed with Ternary Content Addressable Memory (TCAM) (SHARMA et al., 2014). TCAM memories are very fast, however with the limitation of being of high cost and high energy consumption. Currently, OpenFlow switches have between ∼500 and 2500 OpenFlow rules (VISHNOI et al., 2014a) (with high-end switches' supporting more than 100,000 rules), and the practical use of OpenFlow switches memory is vital to operating OpenFlow networks, especially in case of failures.

Besides the memory limitation, the SDN controller has to maintain a consistent view of the network state (SILVA, 2018b; ROTHENBERG et al., 2012). Depending on the approach to fetch information about the network state, the signaling between the switches and controller can play a crucial role in the network communication overhead.

2.3.1 Topology Discovery

One element of the network state is the topology. This subsection provides a brief review of how the SDN controller can fetch the topology information about the SDN network.

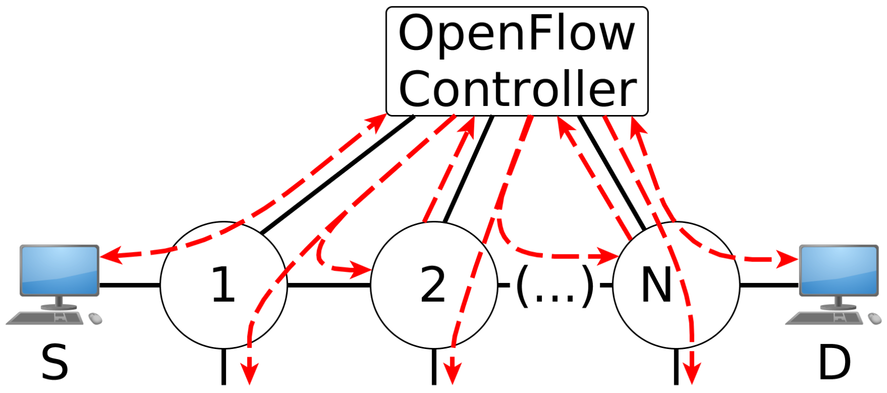


Figure 6 – Topology discovery using flood messages.

The topology discovery process can be reached using flood messages. In Figure 6 represented that processing of discovery of the topology, where the long dash lines represent the control packets from/to OpenFlow switches or hosts. This process uses intensively the OpenFlow flood port (PFAFF et al., 2012), where an incoming packet is distributed to all physical ports except input port and those disabled by Spanning Tree Protocol (STP). The switching loop treatment has to be done for avoiding broadcast storms during the discovery process of the SDN topologies that contain cycles (or loop in the topology).

The SDN controller is responsible for acquiring physical network information, such as links and OpenFlow Switches identifications and features (e.g., supported link rates of each switch port). It also creates a logical abstraction of the representation of the physical topology that can be used appropriately by the other components. Regarding implementation, one discovery protocol, used in Ethernet networks, that provides topology discovery is Link Layer Discovery Protocol (LLDP).

Besides, the controller also monitors the underlying infrastructure to maintenance topology consistency. If a link suffers a failure, it is expected that the controller receives status information from the OpenFlow switch via OFPT_PORT_STATUS message notification (PFAFF et al., 2012). Thus, it is expected that every modification on the topology is communicated to the controller, and then it provides an appropriated treatment to keep the consistency of the topology and network state. The controller can also use either Loss Of Signal (LOS) or Bidirectional Forwarding Detection (BFD)(D. Katz and D. Ward, 2010a) to detect link failure.

2.3.2 Types of flow creation

Differently, from traditional networks, the SDN networks forward packets based on a flow computed by the logic in the SDN controller. For OpenFlow networks, it is an issue to decide where and when a network flow rule has to be installed, once the flow rules consume precious network resources.

Thereby, once the topology is known, the controller can create new flows inside the network. Three approaches can be performed to flow creation: *Reactive*; *Proactive*; and, *Hybrid*. The three algorithms for flow creation are in Figures 7, 8, 9, respectively. The cycles represent OpenFlow switches, and a visual description of the path followed by the first packet from the source node S to destination node D for the three approaches is indicated. The long dashed lines inform the path of the first packet of a new flow and the dotted lines are OpenFlow packets instructing the OpenFlow switches to create the flow from node S to node D.

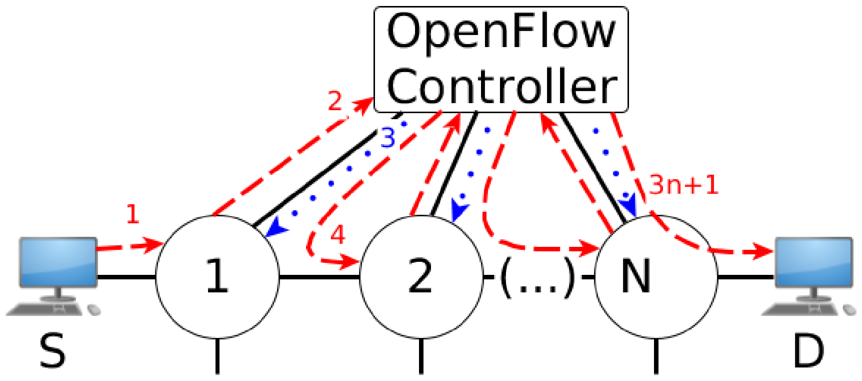


Figure 7 – Reactive flow creation.

The *Reactive* flow creation populated the flows on demand to react upon incoming packet events, OFPT_PACKET_IN messages (PFAFF et al., 2012). When a packet that did not match any rule installed into an OpenFlow Switch, often, the switch enqueues the packet and informs the controller for a new flow creation. Afterward, the controller computes the rules to be associated with the new flow and installs them in the network. Once the rules are installed on the switches, packets are dequeued and forwarded in the network. The freshly-installed rules will then process any subsequent packet of the flow without further intervention of the controller (NGUYEN et al., 2016). Figure 7 represents the *Reactive* flow creation.

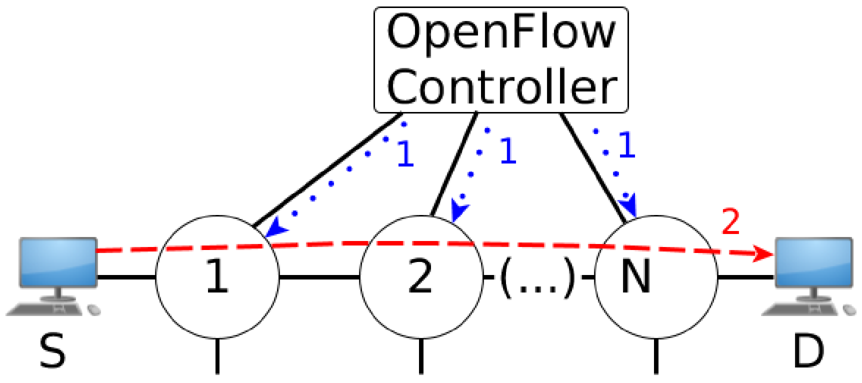


Figure 8 – Proactive flow creation.

The *Proactive* flow creation uses the topology information of an SDN network to connect all previously known source and destination points on the network. Thus, before a packet belonging to a particular flow arrives at the SDN network, the OpenFlow rules already have been installed into the OpenFlow switches for that packet. For this flow creation process, the controller sends control messages to the OpenFlow switches and instructs them to install all flows rules to connect the source and destinations. Figure 8 represents the *Proactive* flow creation scheme.

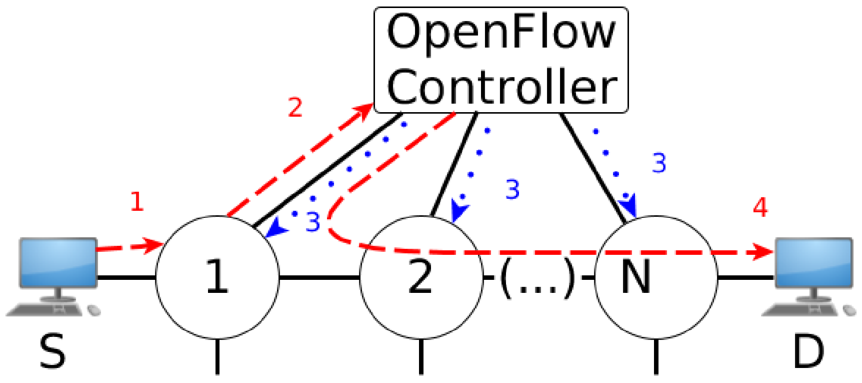


Figure 9 – Hybrid flow creation.

The other way to create flows inside the SDN network is the *Active* flow creation. It exercises bird's eye view of SDN, and once the packet is sent to the OpenFlow controller, it calculates where the destination is and sends the packet to the final data plane element. Besides, it sends OpenFlow control packets to install a new flow for each OpenFlow Switch in the path between the source and destination without further intervention of the controller.

The approach for *Hybrid* flow creation mixes the *Reactive* and *Proactive* flow creation. The difference between *Hybrid* and *Reactive* is that in *Hybrid* approach all OpenFlow Switches receive OpenFlow rules at once, instead of the *Reactive* that requires each OpenFlow Switch, individually, to communicate with the controller to create the new flow.

Also, it is important to highlight the difference between *Hybrid* and *Proactive*. The first does not create rules in advance. It waits for the packet from source to destination arrive into one OpenFlow Switch (or any other event that changes the network state). Then, the switch sends an OpenFlow message to the controller that processes and computes the path from source to destination for that particular packet. Afterward, the controller sent to all switches the OpenFlow messages instructing, those switches, the creation of the new flow. The Figure 9 represents the *Hybrid* flow creation approach.

After a flow is created, it is important that the SDN controller make sure that the network is available. Thus, resilience mechanisms are vital for SDN networks. They assist network operators in reaching high levels of availability.

2.3.3 Multiple Packet-In during flow creation

One instance of SDN concepts is the OpenFlow protocol (KREUTZ et al., 2015) that is specified and managed by the Open Networking Foundation (ONF) (ONF, 2017). ONF already defined a couple of versions of the OpenFlow, and every time a new version of OpenFlow is released, new network headers protocols are embraced and included in the protocol specification. However, not all management actions and functionalities can be predicted and solved at the protocol level (BOSSHART et al., 2014). Thus, the OpenFlow controller has to be used to manage those exceptions.

Based on the OpenFlow protocol specification, the controller enables the programmability of OpenFlow switches by installing predefined actions to be executed in the switches. However, whether any predefined actions in the switches are not enough to handle the incoming packets, then the switches can transfer those packets to the controller. This mechanism of transferring packets from switches to the controller is called *Packet-In*, and it allows the controller to apply custom actions to packets that were not predicted in the OpenFlow specification.

One of the problems of *Packet-In* packets is the concentration of messages (ANDO; NAKAO, 2016) sent from switches to the controller. Figure 10 depicts a typical case of incoming multiple *Packet-In* to the controller. When a transmission is starting, the switch receives those packets in its interface port. If there is no matching in the already installed rules, the default rule usually is to send the packet to the controller (using OFPP_CONTROLLER port as described in the OpenFlow specification (ONF, 2017)). Thus, the incoming packets are encapsulated in an OpenFlow message and sent to the controller as a *Packet-In*.

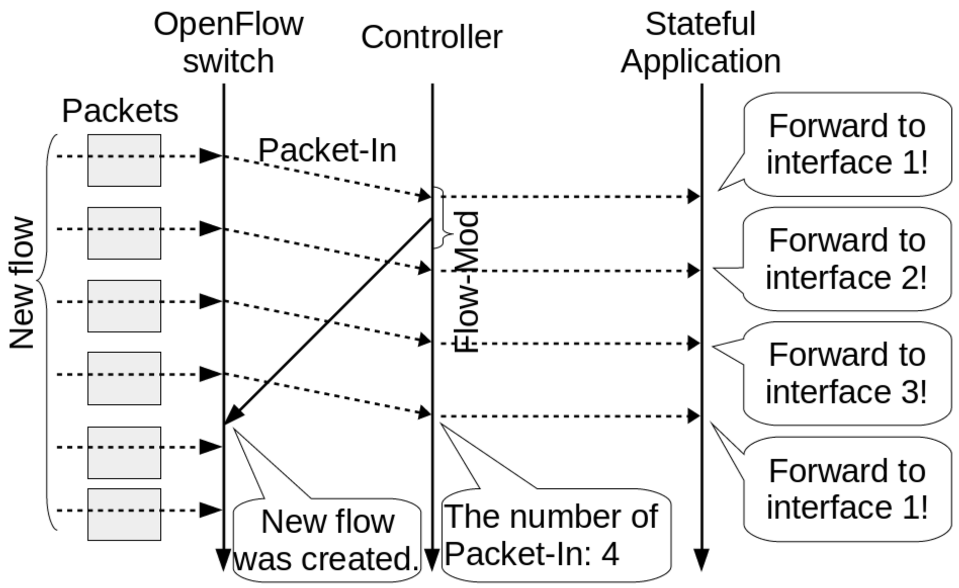


Figure 10 – Several *Packet-In* packets sent to the OpenFlow controller for the creation of new flow.

The quantity of the packets received at the interface of the switch and encapsulated in a *Packet-In* packet is indicated by the "max_len" field of the *Output* action (an action that can be used to forward packets to the controller (ONF, 2017)). Setting the appropriate value of "max_len" can decrease the bandwidth required between switches and controller, once not all the information in the packet received from the network is used in the management of flows, especially and often in the flow creation process. Thus, the header of the packet is analyzed, and the decision about how to handle the *Packet-In* is performed. Finally, the switch will stop sending *Packet-In* packets when the controller instructs the switch how to handle that traffic, using a *Flow-Mod* packet.

In Figure 10, the switch sent 4 (four) *Packet-In* packets to the controller, before the rule for creating a new flow in the switch was being installed. At the controller, the received packets also generate *Packet-In* events. Those events are distributed to applications that previously registered in the controller its intention of receiving *Packet-In* events. Whether the controller passes all *Packet-In* events, the burst of packets can alter the state of the application and its logic. And that is why the controller has to provide a consistent view of the network state to their applications.

The *Stateful Application* in Figure 10 received all 4 *Packet-In* events from the controller and changed its state four times for a new flow, instead of changing state only once. The properly control of the quantity of information sent from the controller to a stateful application requires a modification in the switch or controller side.

2.4 RESILIENCE MECHANISMS

It is essential to have a minimal guarantee that the network will remain available and operational, even when part of the network goes down (e.g., caused by a link failure). For OpenFlow networks, the resilient mechanisms are approaches that seek to bring minimal guarantees for the network connectivity.

Thus, this section presents the challenge of managing the number of rules required in a resilience mechanism for OpenFlow network, once the switches have a limit number of available rules that can be used, and the decoupling between data and control plane plays a critical role in the network availability when compared against the traditional network. Also, this Section starts with a presentation about resilience mechanisms approaches. Then, it compiles a few recent works and approaches in the field for the resilience mechanisms. Finally, this section concludes by describing the dynamics of resilience mechanisms extracted from the related works.

2.4.1 Resilience mechanisms approaches

Resilience mechanisms are vital for OpenFlow networks. They assist network operators in reaching high levels of availability. However, a good resilience mechanism for OpenFlow networks is challenging because there must be a trade-off between rule installation and the signaling overhead between OpenFlow switches and the OpenFlow controller (SILVA, 2017; SILVA; DIAS; SADOK, 2017). The resilience mechanisms have been classified into three categories (SILVA, 2017): restoration approach (or *Reactive*); protection approach (or *Proactive*) (NGUYEN et al., 2016; FERNANDEZ, 2013; ADRICHEM; ASTEN; KUIPERS, 2014; AKYILDIZ et al., 2014); and hybrid approach (or *Active*) (SILVA, 2017).

Restoration approaches create rules on demand to react to failure events. For example, whenever an a link failure occurs, the controller reacts by signaling the data plane elements to restore the broken flows affected by the failure. For OpenFlow networks, restoration approaches are similar to the usual process of flow creation (see Section 2.3.2), where the difference is the cause: for a restoration approach a failure starts the flow creation; and, in the flow creation, network demand initiates the process.

In a protection approach, rules are populated in advance, when backup paths are configured and installed before failures occur. In other words, a protected flow is created in the network before a failure occurs or any packet belonging to the path arrives at an OpenFlow switch port.

Instead of restoration and protection approaches, a third type in the classification of resilience mechanisms can be used, as well discussed by Silva (SILVA, 2017): the *Hybrid*, that is a solution that merges restoration and protection. The *Hybrid* approach optimizes the control of network traffic, signaling, and memory utilization by exploring the bird's eye-view of SDN. This is reachable with the monitoring state of the network searching to mitigate the occurrence of a failure.

2.4.2 Dynamic of resilience mechanisms

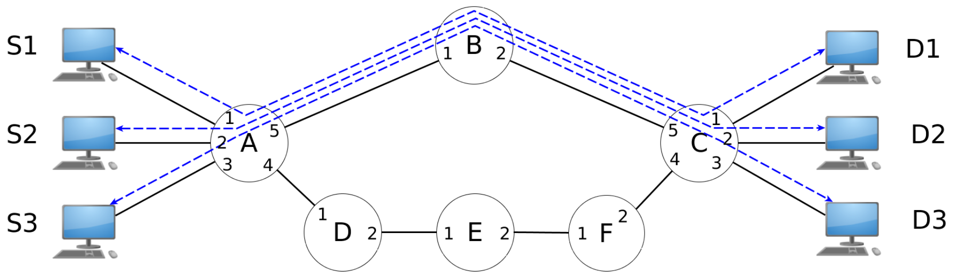


Figure 11 – Flows created using SFC

The resilience mechanisms take actions when a failure event occurs, and those actions can be previously made (e.g., protection approach) or made after the failure event (restoration). The dynamic of each resilience mechanism depends on the strategy adopted for creating flows into the network. The SFC is the most trivial way to create flows in an OpenFlow network. It creates a rule for each flow installed into each OpenFlow switch in the path between the source to the destination. The Figure 11 depicts three flows from Sources (S1, S2, and S3) to Destination hosts (D1, D2 and D3) using SFC. Thus, taking into consideration the SFC, the three types of resilience mechanisms can be depicted.

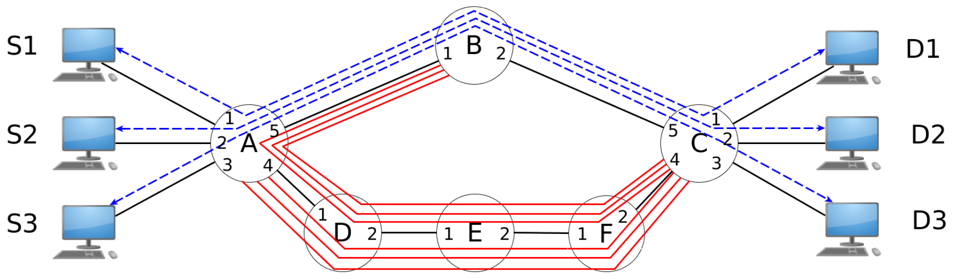


Figure 12 – Additional rules of PP

To protect the path [A, B, C], a resilience mechanism can create backup rules to avoid unavailability when a primary path suffers a link failure. In the case of the path [A, B, C], the affected links can be linked A-B or B-C. A simple approach would be to install OpenFlow rules to protect the link A-B and other rules to protect the flows from a link B-C failure. This logic is presented in Figure 12, where the PP applies OpenFlow rules to protect the path [A, B, C]. Thus, if link A-B or B-C suffers a failure, the flows from hosts S1, S2, and S3 still reach destination hosts D1, D2, and D3 by using the additional flows installed in advance for the protection scheme. Besides, no signaling between the controller and OpenFlow switches is needed. Nonetheless, extra rules are required to compose an operational solution (as indicated by the paths [B, A, D, E, F, C] and [A, D, E, F, C] in Figure 12).

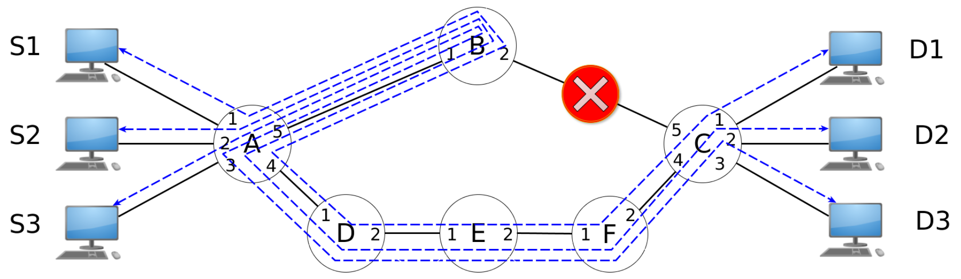


Figure 13 – Application of LR after a failed link

Instead of the protection mechanism, a network operator can opt for a restoration approach, where new flows are installed after failure. Figure 13 depicts the operation of LR. LR is indicated when the computation path time is not a bottleneck of the resilience scheme to restore the flows (network with a few number of nodes and links) (SILVA, 2017). Thus, when a link failure occurs, the OpenFlow controller acquires the information of what nodes have involved and reconstructs the broken path between those nodes. If the packets belong to the affected flows, they need to be sent back through the same input port of an OpenFlow switch; the OpenFlow specification has a special port for this case. The OFPP_IN_PORT is a virtual port that has to be explicitly set in OpenFlow rules for a switch send a packet back to the same input port (PFAFF et al., 2012).

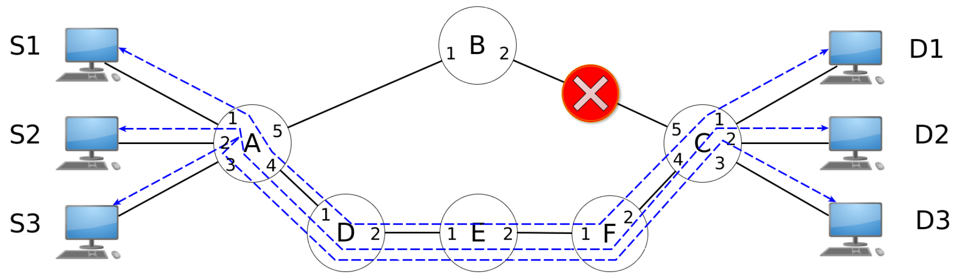


Figure 14 – Application of PR after a failed link

Another restoration mechanism is the PR (see Figure 14), which the work of Sharma et al. (SHARMA et al., 2011) uses. Different from LR, PR exercises the global view of SDN networks and computes the best path (often using the shortest path algorithm) for each flow affected by the broken link. Note that under certain circumstances, LR and PR can produce the same outcome. In the topology used as the example (see Figure 11), if link A-B suffers a failure, the recovery mechanism of LR and PR will produce the same final network state because Node A is the disjoint point between the primary and backup path ([A, B, C], [A, D, E, F, C], respectively), however, if link B-C suffers a failure, the outcome of LR and PR application will be different.

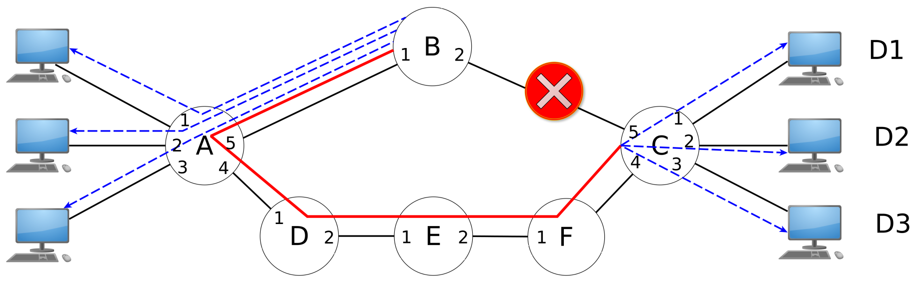


Figure 15 – After link B-C failure and application of LFR.

To reduce signaling between controller and OpenFlow switches, as well as the number of flow entries installed, a resilience recovery strategy can employ an aggregation scheme. The aggregation scheme can use the Internet Protocol (IP) network mask or tunnels. The application of LFR is presented in Figure 15. Thus, instead of treating every flow individually (as PP and PR), LFR aggregates all flows into one "big" flow that makes all packets affected by the failure be redirected using this "big" flow. An example of work that applies the LFR concept is in Zhang et al. (ZHANG et al., 2016). Therefore, this subsection exemplifiers and depicts the resilience mechanisms extracted from related works.

2.5 A BRIEF REVIEW OF BGP

The Internet is a collection of tens of thousands of independently operated networks, simply called Autonomous System (AS)s. An AS can be an Internet Service Provider (ISP), a campus, a content provider or any other independently operated networks. Thus, to carry traffic from one AS to another, an AS requires two types of routing system: intra-domain routing; and inter-domain routing.

Intra-domain routing is the process routing network traffic inside any single autonomous system in as a fast, effective and reliable way as possible. Examples of protocols for intra-domain routing are OSPF, Routing Information Protocol (RIP), and internal BGP (iBGP). However, for inter-domain routing, the focus is on applying routing policies and distributing routing reachability information among ASs. For the inter-domain routing system, the external BGP (eBGP) is the standard protocol of the Internet.

2.5.1 BGP Control Messages

The BGP exchanges messages between ASs using the Transmission Control Protocol (TCP). The TCP port number 179 is allocated exclusively for the BGP protocol. The advantage of using TCP is avoiding the BGP control plane having to manage message delivery and flow control between BGP speakers (or peers), which simplified the BGP design. Thus, a BGP peer has a reliable way to exchange Network Layer Reachability Information (NLRI) and compose the Routing Information Base (RIB).

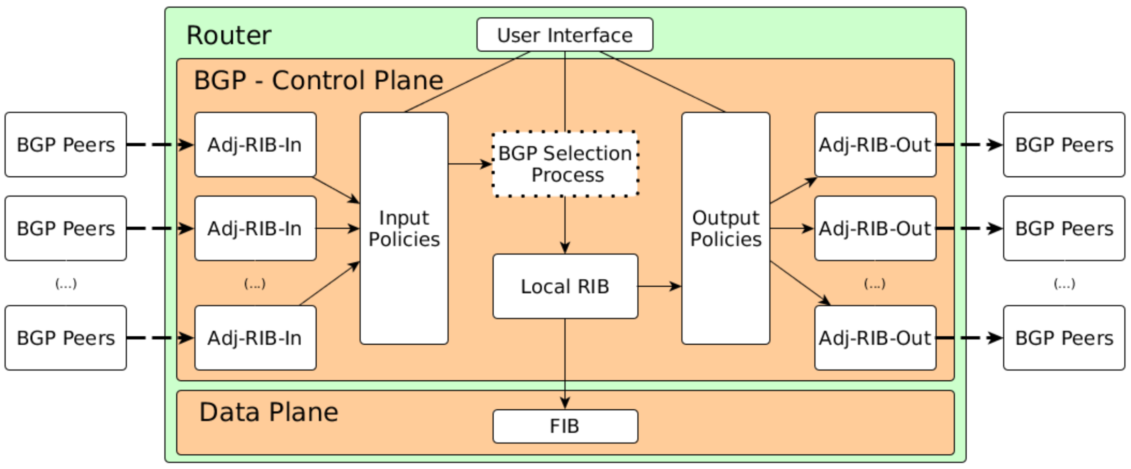


Figure 16 – Model of the BGP control plane.

Each BGP peer needs to be manually configured with a set of parameters to enable a BGP connection (YANNUZZI; MASIP-BRUIN; BONAVENTURE, 2005; REKHTER; LI; HARES, 2006). There are four types of BGP control messages to be exchanged after a TCP connection is established between two BGP peers (REKHTER; LI; HARES, 2006):

- OPEN message: sent to open a BGP session and to verify the connection's parameters;
- UPDATE message: to transfer network reachability information by advertising and withdrawing routes;
- KEEPALIVE message: periodically sent to ensure that the connection between the peers is still reachable;
- NOTIFICATION message: used in response to special or error conditions.

Additionally, the BGP is a path vector protocol, and it uses a sequence of Autonomous System Number (ASN)s for characterizing the path. Thereby, when the BGP updates travel through different ASs, the BGP routers prepend their ASN to the AS-Path attribute. The AS-Path attribute of BGP carries the ASNs that a given network prefix traversed. If a router receives a BGP update message and detects its own ASN in the AS-Path attribute, then the router will ignore the update because it is a routing loop once the message has already passed through the AS.

Each BGP update message contains NLRI and BGP information to apply the routing process. For the BGP, the RIB is constructed through the BGP decision process executed in the BGP control plane.

2.5.2 BGP Control Plane

The BGP Control Plane is the place where decisions about how to handle the network traffic are made and where the traffic is sent. Figure 16 presents an abstraction of a BGP router in which the BGP control plane is modeled.

The responsibilities of the control plane are managing network traffic, setting system configuration, and the exchange of routing information. To set up the control plane, an *User Interface* is provided and is often the command line of the network appliance.

The BGP control plane uses UPDATE messages to exchange Network Layer Reachability Information (NLRI) with other routers to create the topology view of the network state and to build the routing table. The BGP configuration reflects the business model of the organization that executes it. The main goal of the BGP is to reflect the desired inter-domain routing policy of a particular AS. The BGP routing decision process applies the filtering and export for a given AS to learn routes from its neighbor and re-advertise them. The final goal of this BGP decision process defines one "best" route per prefix.

Once a TCP connection of the BGP is established, and the route information is exchanged, the Routing Information Base (RIB) is filled. The adjacency table *Adj-RIB-In* is the input for the BGP routing decision process, and it is where the attribute manipulation is also carried out (such as checking *communities* values). Then, the import policy/filter determines which routes are acceptable from each BGP peer. This generates a new database called the Local Routing Information Base *Local RIB* that stores all acceptable routes learned from all BGP peers plus the internal routes, which are routes that belong to the AS and were learned through intra-domain routing protocols. Thereby, to construct the *Local RIB*, the control plane compiles information of the BGP, and all other routing protocols (for example, injected by OSPF and RIP) and static routes enter by the network operator.

Table 1 – The BGP path selection decision process

Priority	Criterion
1	Prefer route with highest Local preference
2	Path originated by a local router
3	Path with shorter AS-path length
4	Path with lowest origin code
5	Lower MED values for routes from the same neighboring AS
6	Prefer routes learned from external eBGP rather than iBGP
7	Prefer the path with the lowest IGP metric to the BGP next hop
(...)	Other BGP tie-break

Once the routes are learned, the BGP control plane has to decide a single "best" next hop of each destination. First, if the prefixes learned are unique, then they are installed in the forwarding table of the BGP appliance, and the longest prefix matching is applied. However, if some prefixes have the same subnet mask, but different next hops, then the BGP uses its tie-break algorithm to decide what is the next hop for a given network prefix.

Table 1 presents that algorithm, where the priority indicates the sequence of the criterion used in this process. The first criterion for the BGP route selection process is preferred high *Local preference* values, which is a numerical value that a network operator, in the local AS, can assign to a particular route. If the *Local preference* value is equal for more than one route, then the BGP will follow another tie-break criterion. This way, the BGP decision process selects just one *best* route towards each destination.

The finalization of the BGP decision process is the export policy/filter that determines which routes can be sent to each *Peer*. Again, the attribute manipulation is also applied on the routes, and then, they have stored in the *Adj-RIB-Out* database. Then, other *Peers* can learn the routes available for them. All the routing information is then processed to update the routing tables of the BGP router. The control plane functions do not process each packet of the device. This responsibility belongs to the data plane elements.

The *Data Plane* is also known as the *Forwarding Plane*. It forwards network traffic to the next hop along the route to the selected destination following the logic given by the control plane. The Forwarding Information Base (FIB) stores the information of which interface packets should be forward to. Thus, the *Data Plane* elements (e.g., routers or switches) use the FIB to forward incoming and outgoing packets or frames to one of its interfaces. For a BGP router, all packets pass through the *Data Plane* of the router.

2.5.3 Monitoring Prefixes

The traditional BGP monitoring can be simplified as collecting data from *Adj-RIB-In*, where it stores the received prefixes from the neighbors of a router, *Adj-RIB-Out*, where it stores the advertised prefixes, and the *Local-RIB*, which is the selected, used and installed prefixes for a router. Some routers can store the prefixes of the *Adj-RIB-In* before and after the application of the routing policies.

Hence, the *Adj-RIB-In (Pre-Policy)* stores all received prefixes from a Peer and the *Adj-RIB-In (Post-Policy)* is yield after those prefixes pass for filters or any modification of BGP attributes (e.g., Communities). The main benefit of this capability is to reconfigure and activate BGP routing tables without tearing down existing peering sessions, whenever the routing policy changes due to a configuration change, and the drawback are more memory required to store all received routes before and after the application of routing policies. The *Adj-RIB-Out (Pre-Policy)* and *Adj-RIB-Out (Post-Policy)* work as the same way as the *Adj-RIB-In (Pre-Policy)* and *Adj-RIB-In (Post-Policy)*, with the difference of being used for advertising prefixes to the Peers.

Thus, the easy way to get the routing information that is actually being used by a router is accessing its *Local-RIB*. However, to access the *Local-RIB* of a router, it

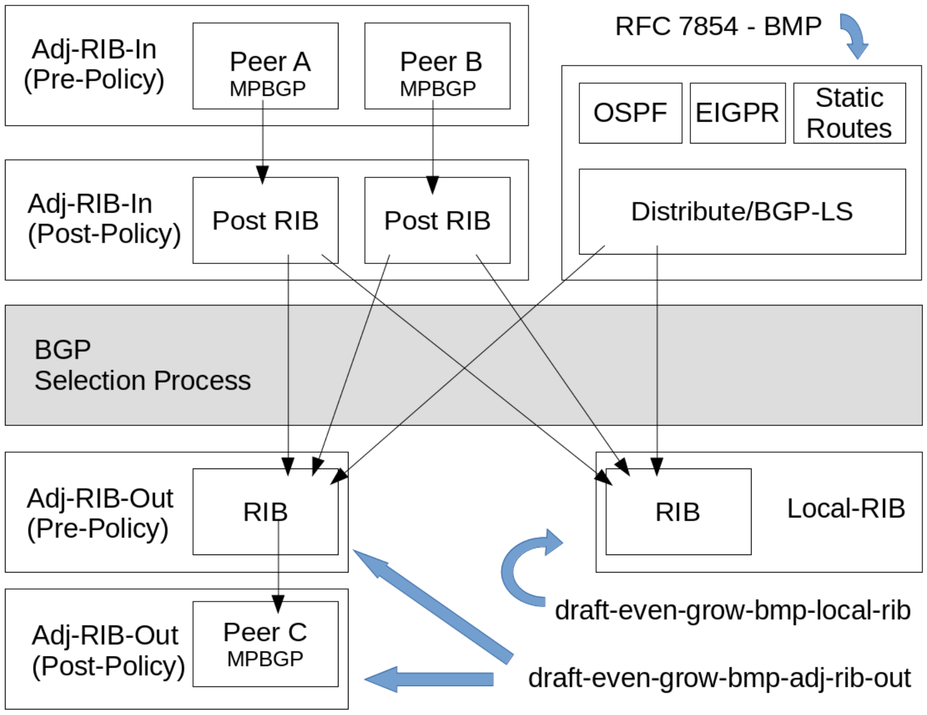


Figure 17 – Monitoring points in BGP.

requires a peering BGP session to exchange (advertise or receive) prefixes. To overcome that limitation, BGP Monitoring Protocol (BMP) can be used to execute such BGP collection (SCUDDER; FERNANDO; STUART, 2016) and other type of routing information, e.g. IPv4, IPv6 or *Link State* (locally originated). Besides, new RFCs drafts also will incorporate new monitoring points in BGP, for example, *draft-even-grow-bmp-local-rib* and *draft-even-grow-bmp-adj-rib-out* that will be responsible for getting *Local-RIB*, and *Adj-RIB-Out (Pre-Policy)* and *Adj-RIB-Out (Post-Policy)*, respectively.

Therefore, instead of just focus on one router at each time, new network tools are allowing collecting routing information from multiple routers. Thus, the idea of monitoring BGP is getting that information from more than one Peer. The Figure 17 depicts the points where can BGP collection can be performed with the Peers executing the Multiprotocol Extensions for BGP (MPBGP) (which is an extension to the BGP that enables multiple types of address families to be advertised simultaneous during a BGP session).

2.6 STATE-OF-THE-ART TECHNIQUES FOR INBOUND TRAFFIC CONTROL WITH BGP

The current state-of-the-art techniques for inbound traffic control is based on applying the prefix disaggregation, AS path prepending or the use of BGP communities (requires standard policies and configured the system to be successful) (DEUS; CARVALHO; LEITE, 2014). Thus, depending on the AS goals, it can apply those TE techniques to reach the desired traffic network load through its inter-domain links.

Before delving into the state-of-the-art techniques for inbound traffic control with BGP, a brief description of AS relationships is required. Then, a process for apply BGP inbound traffic engineering with BGP is depicted.

2.6.1 AS relationships

AS relationships determine how routing policies must be set up to absolve business constraints, agreements, and requirements. However, not all ASs are of the same types. A stub AS (or non-transit AS) typically refers to an AS that has a connection to the Internet through at least one other AS, it can use a public or private ASN to establish peer relationship between its neighbors, and the network traffic is a source or sink. Further, if the stub AS uses just one link to the other AS, it is classified as a single-homed connection. Otherwise, it is a multi-homed AS (using multiple links to the other AS). The Figure 18 presents a scheme for single homed and multi-homed stub AS.

The other type of AS is a transit AS, and that is an AS that allows connectivity of other ASs through itself. In Figure 18, the ISP are transit ASs because it carries network traffic from an AS to the Internet and from the Internet to the AS (which can be a stub or transit AS). Therefore, depending on connectivity and operational policies between ASs, a AS can play a role of stub or transit AS.

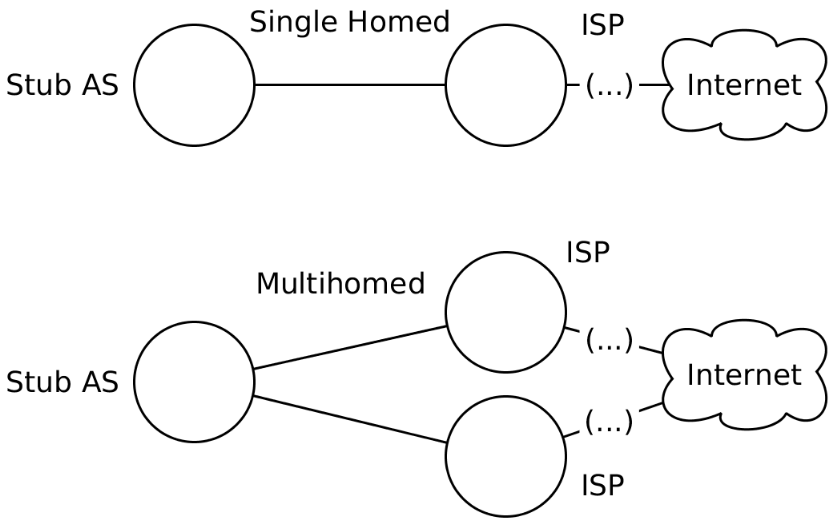


Figure 18 – Single homed and multihomed scheme.

Regarding ASs relationships, they may be complex and are usually not exposed to public access. However, it is possible to abstract the relationships among ASs by simplifying them into one of the two types:

- Customer-to-provider: the relationship where the AS provider (or upstream) is paid by the customer AS (or downstream) to carry customer network traffic from or to other networks;

- Peer-to-peer: the ASs involved agree to share the costs of the connectivity among them, and then, all traffic between them is free of charge.

The customer-to-provider relationship ensures that AS providers are paid by the customer regardless of the direction in which traffic flows. Thus, for a particular AS, it is preferable to first route traffic for customer links, then peer-to-peer links and finally to provider links (when it is acceptable). Because the volume of traffic matters to define a transit AS profit, it is mandatory to control how the inbound and outbound traffic is routed. Besides, for non-transit ASs' (stub ASs) control, the inbound traffic is mandatory to reach the full utilization of their inter-domain links.

2.6.2 Traffic Engineering with BGP

Control of the outbound network traffic is an easy task with BGP. Since an AS controls the decision process on its BGP routes, it can select each best path to reach a particular destination through its peers. The AS can rely on the *Local preference* attribute, for example. By assigning appropriate values, the AS indicates which route should be considered as the best route to the BGP routers (LUDWIG, 2009).

However, it is not a simple task to control inbound network traffic because it requires techniques to be applied that do not guarantee its effectiveness (Ning Wang, Kin Hon Ho, George Pavlou, 2008). A network operator can use the following techniques to influence the incoming network traffic with the BGP (CARDONA et al., 2016): Multi-Exit Discriminator (MED); Selective announcements; AS path prepending; and, Community.

Multiple Exit Discriminator (Multi-Exit Discriminator (MED)), or Multi-Exit Discriminator, is a BGP attribute that provides a "hint" to external neighbor routers about the preferred path into an AS that has multiple entry points. This suggestion can, or not, be acceptable for the neighbors. Indeed, MED is only useful for informs neighbors peers (SHAO et al., 2015a) because MED is a non-transitive attribute (see Table 1), which means that it is not propagated throughout the Internet, but only to neighbors ASs.

The idea behind MED is to override the "hot potato routing" behavior, which is the practice of passing traffic off to another autonomous system as quickly as possible. Thus, because in the BGP path selection decision process the lowest value of MED is verified first than the lowest IGP metric for defining the BGP next hop, MED is used to indicated how the traffic must enter a given AS.

Another traffic engineering technique using BGP is the *Selective announcements* (or *Selective advertisement*) that relies on careful advertisements of different routes through different links. *Single Homed* AS can not explore the full potential of this technique, once only one link is available. Besides, this technique is indicated to avoid performance problems of transit providers or to balance incoming traffic.

Moreover, inbound traffic policies are applied by influencing the best path selection of external BGP speakers. Thus and because more-specific prefix always wins, independent of any configured BGP attribute set into the BGP UPDATE message, all BGP traffic engineering efforts becomes pretty useless when selective announcements are applied.

By default, the BGP prefers the route with the shortest BGP AS-Path attribute length when two or more routes exist to reach a particular prefix. *AS-path prepending* technique occurs by increasing the AS Path length of a particular route. The goal is trying to change the way that the other ASs selects the best routes and consequently, how the other AS traffic is handled.

The Community is a BGP attribute and can be used as "tags" associated with advertisement prefixes. The main goal of a BGP Community is to identify routes to be applied in the routing policies. Thus, appended pre-arranged communities can be used to influence path selection of other ASs if the neighbors consider those communities in their routing decision process.

Furthermore, the BGP Community value has meaning and application for those ASs that include the BGP Community into the routing policies of their domains. Although BGP Community can be set as any value of 32 bits, some values have an administrative assignment, e.g., *Well-Know Communities*. *Well-Know Communities* are Communities that have defined values with a given mean and published in the RFC-1997 (CHANDRA; TRAINA, 1996). For example, No-export community (with the value 0xFFFFFF01) means do not export to any BGP neighbor at all, and No-advertise community (value 0xFFFFFF02) informs to the neighbor routers not to advertise a prefix to any BGP neighbors (iBGP or eBGP).

In general, network operators can only rely on the manipulation of the BGP's attributes to apply any traffic engineering technique. However, for inbound traffic engineering, those approaches only try to *influence* the routing decisions of external ASs, to obtain their desired inbound traffic distribution (SHAO et al., 2015a).

Nonetheless, because each AS selects preferred routes based on its policies, usually the BGP inbound techniques lead to a trial-and-error process with no guarantees of success. Therefore, the agreements, contracts, and AS relationships dictate how the configuration of the BGP should be performed for a given AS.

2.7 FINAL REMARKS

This Chapter introduced the fundamentals concepts that will be used in the remaining of this Thesis, such as SDN, OpenFlow, and BGP. For a more detailed explanation about those topics, see the References for this work.

3 RELATED WORKS THAT EVOLVE THE INTER-DOMAIN ROUTING

The contribution of this Chapter is given an original classification of previous proposals to evolve the control plane of the inter-domain routing.

3.1 ISSUES IN EVOLVING THE INTER-DOMAIN ROUTING CONTROL PLANE

There has been a myriad of publications on the inter-domain routing field, and despite the success of the Internet, it still has some crucial issues and research challenges regarding its operation and design that need to be addressed. This section presents the issues described in related works that impact the evolution of the control plane inter-domain routing systems. Each issue is explained and systematized into the following types: ossification; backward compatibility; distributed configuration; complexity routing policies; coordination among ASs; traffic engineering.

3.1.1 Ossification

The Internet architecture is no longer a coherent whole. It has various components such as transport protocols, routing mechanisms, firewalls, load balancers, security mechanisms and other middle-boxes. Furthermore, the BGP is one of the most successful protocols on the Internet, and although many researchers have indicated issues related to the BGP since its inception (e.g., lack of end-to-end service guarantees, long convergence time, security issues (KEVIN et al., 2010)), practical deployments for new network architecture and protocols in the inter-domain environment are difficult to achieve.

The major problem for evolving the inter-domain routing system is called "ossification", where the architecture becomes very dependent on the protocol and new features are inherently difficult to introduce into the network (CHOWDHURY; BOUTABA, 2010; ESTEVES; GRANVILLE; BOUTABA, 2013; CHOWDHURY; BOUTABA, 2009). For the BGP, the ossification is due to economic reasons and the fact that backward compatibility has to be assured since there is no flag day to switch to a new architecture. Thus, all BGP appliances have to execute the same version of the protocol to operate appropriately, or anomalies in the network may occur (e.g., BGP black holes).

In other words, the BGP is an inflexible protocol as a consequence of the control and data plane being embedded into the network hardware. The dependency between a specific infrastructure and the hardware required creates a barrier to architectural innovation in the inter-domain routing. Hence, a new feature for the BGP has to have a minimum integration within heterogeneous networks and interoperate through different administrative domains (e.g., the Internet).

3.1.2 Backward compatibility

A new feature for the BGP has to have a minimum integration within heterogeneous networks and operate through different administrative domains (e.g., the Internet). Those requirements frustrated new proposals to evolve the BGP and the ecosystem of inter-domain routing (SILVA; SADOK, 2017). One of the main reasons for this is new proposals that have to be maintained with the infrastructure already built, and ASs will not rely on immature technologies that do not prove to be profitable or are not operable with other ASs on which they depend (THAI; De Oliveira, 2013).

Thus, the difficult-to-change architecture of the Internet creates a mandatory requirement for integration among different proposals and the BGP. The backward compatibility imposes a limitation to "clean-slate" proposals. Brand new and incompatible BGP proposals are considered unrealistic (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015). In fact, in the more than two decades since the first version of the BGP protocol (Y. Rekhter, T.J. Watson, 1995), much scientific and industrial effort have been made to overcome the BGP limitations and failed to make it a flexible protocol. The backward compatibility with the BGP for the new approaches in the inter-domain environment continues to be a mandatory requirement.

However, a major criticism of the BGP is that it is not suitable for the next generation of inter-domain networks (THAI; De Oliveira, 2013). The main argument to sustain that claim is any new proposal for inter-domain routing has to be part of traditional networks to operate on the Internet and those proposals will be restricted by the limitations of the BGP.

Moreover, the BGP is a widely-adopted protocol used in the infrastructure of the major part of the Internet. It is an inflexible protocol, and it is difficult to incorporate new changes into the network infrastructure. Therefore, the control plane of the inter-domain routing must be evolved instead of receiving a revolution. Evolving the Internet control plane is not an easy task because to be practical and useful, it is almost mandatory to have backward compatibility with the current technologies (YANNUZZI; MASIP-BRUIN; BONAVENTURE, 2005).

3.1.3 Complexity introduced by the distributed configuration

The traditional approach for the inter-domain routing is through a fully-distributed path selection computation in IP routers and that limits the capacity of individual ASs in terms of the management scalability of path selection (FEAMSTER et al., 2004). Besides, a network configuration that changes frequently, to suit the business and organization's desires, is a counterproductive challenge. Thus, the big challenge of distributed configuration concerns the complexity of keeping a consistent network policy through all routers. The complexity of managing the configuration in distributed devices to make them operate with the acceptable network behavior is a hard task. Additionally, network management is a complex activity due to it requiring the enforcement of the high-level administration network policies in the network.

Hence, a network operator needs to individually visit each network device and perform low-level instructions/commands (often vendor-specific) on them. Depending on the number of network devices, carrying out such procedures is slow and error-prone (KREUTZ et al., 2015), and a local network error can affect all the other networks. For example, for the famous case of the *Pakistan Telecom* incident BGP misconfiguration, in which the access to *YouTube* in that country was restricted, the company advertised an unauthorized prefix causing many ASs to lose access to the site. Therefore, the complexity introduced by a distributed configuration can affect not only the specific network where a misconfiguration occurs, but also the whole inter-domain routing environment.

3.1.4 Conflicts and uncertainty in inter-domain routing policies

The BGP incorporated routing and policy requirements into one seamless protocol. For inter-domain routing, each BGP router has its own view of the network state and applies routing policies based on its local configuration. This led to the concept of the routing policy, which can be understood as how routing decisions are made to compose the network reachability state (AL-MUSAWI; BRANCH; ARMITAGE, 2016). The difference between routing and policy control is that the first is more related to an engineering effort and the second is strictly about business (THAI; De Oliveira, 2013). Both functions are embedded into the BGP current protocol version, despite the fact that its original design was only for routing purposes.

Moreover, different types of ASs deploy different types of routing policies into their domains, and that diversity of policies can eventually lead to conflicted ones. An example is the application of inbound traffic engineering techniques using the BGP techniques. The main idea of an AS to control its inbound traffic is to fulfil the AS' businesses interests. However, applying the BGP mechanisms for inbound traffic control (such as *Selective advertisement* or *AS-path prepending* (Ning Wang, Kin Hon Ho, George Pavlou, 2008)) does not guarantee the effectiveness of the desirable results.

In fact, the BGP techniques try to influence the routing decisions of external ASs, in order to obtain their desired inbound traffic distribution. Hence, a network operator that uses those approaches leads to a trial-and-error process, since other ASs can easily ignore or withdraw the BGP *suggestions* coming through BGP update messages. Indeed, the problem of inter-domain routing traffic engineering can be seen as a conflicting one, in which the interactions between ASs can be modeled as game theory and nonlinear programming (Ning Wang, Kin Hon Ho, George Pavlou, 2008).

Another issue due to conflicts and uncertainty in inter-domain routing policies is to discover what the network traffic distribution of the inter-domain environment will be when policies between domains have to change. The prediction of how much bandwidth will be consumed or how the traffic will be distributed when a domain desires to change its inter-domain routing policies for a given prefix are difficult to estimate (DEUS; CARVALHO; LEITE, 2015). Therefore, proposals to evolve the control plane of inter-domain routing have to provide schemes, mechanisms or protocols to overcome the BGP limitations to express new and refined policies that avoid conflicts and uncertainty in the inter-domain routing policies.

3.1.5 Coordination among ASs

The inter-domain routing on the Internet is characterized by the use of best efforts for carrying network traffic, where thousands of distributed and connected ASs make their own decisions about how traffic should be routed and forwarded. Besides, it is assumed that packets can be lost and suffer delays, but the overall network will still to be operational and functional. Although the resilience requirement is one of the main concerns, other attributes are desirable for the inter-domain routing environment. For example, some network applications, such as Voice over Internet Protocol (VoIP), require guarantees from the network for its appropriate utilization, and to provide end-to-end requirements, it is fundamental to have some level of coordination among ASs.

In the inter-domain routing, the coordination between ASs depends on the technology used. In general, each AS provides the manual configuration of BGP router peers to exchange NLRI. Those configurations are aligned with the routing policies and describe what is or not allowed to be advertised between neighbor ASs. Thus, the coordination among domains is often restricted to ASs that have a previous business relationship or some type of an agreement, such as a customer-to-provider relationship. For example, if a stub AS requests the verification of the BGP configuration to a transit AS that does not have any formal relationship with the requesting AS, it is expected that the request may not be fulfilled by the transit AS (often ignored).

One motivation for improving the coordination among ASs for inter-domain routing is security (KEVIN et al., 2010, 2010; NARAYANAN, 2009). For example, the difficult to enforce and track the origin of ASs that advertise a given IP prefix is a security issue that threatens the inter-domain routing system. Relying on the BGP attributes (e.g., AS-Path attribute) for such tasks has demonstrated its ineffectiveness over the last few decades. Thereby, deliberate network attacks (e.g., Man-In-The-Middle attacks (MITM) or Distributed Denial of Service (DDoS) (SOMANI et al., 2015) attacks), misconfigurations or even errors (ALSHAMRANI; GHITA, 2016) in a BGP router can cause and affect connectivity problems on the whole Internet. Those issues occur because the standard control plane of the BGP was not designed to provide security mechanisms for ASs' trust in the NLRI advertised on the Internet.

3.1.6 Traffic engineering in the inter-domain

The traffic engineering inside an AS is performed using intra-domain routing systems, which are the processes of routing inside any single AS with the purpose of delivering the packets as efficiently as possible. To manage intra-domain routing, there are a couple of routing protocols, for example OSPF, RIP, Enhanced Interior Gateway Routing Protocol (EIGRP), Intermediate System to Intermediate System (IS-IS). Each protocol has its own direct mechanisms to reach the goal of applying traffic engineering techniques. However, inter-domain routing that fulfills traffic engineering requirements (e.g., Quality of Service (QoS)) is more tricky.

The major design focus of the BGP was a routing protocol capable of applying routing policies and scale when exchanging connectivity information, once scalability is a paramount requirement for inter-domain routing protocols. The BGP has proven its scalability and efficiency to impose routing policies over years of deployment. Nonetheless, ASs that use exclusively the BGP for inter-domain routing cannot receive other types of routing information other than connectivity. Besides, due to the fact that the BGP is designed to use only one best route to each destination entry (see Section 2.5.2), the control plane has limited connectivity information for routing in a more optimized route selection when it is possible.

For example, due to the lack of direct mechanisms to explore the path diversity of the Internet, ASs have been seeking to improve resilience requirements for their domains, and a common technique to reach that goal is the ASs splitting their prefixes and advertising them for different paths. The side effect of this technique is the Internet routing table growth, which affects the Internet scalability. The Figure 19 presents the Internet routing table size growth over almost the last two decades, where the number of unique public ASN has a line growing over the time, see Figure 20.

As is depicted in Figure 21, the Internet is composed mostly of stub ASs. The exhaustion of Internet Protocol version 4 (IPv4) and the application of BGP techniques to provide independence addressing, load balancing and fail-over techniques for multi-homing ASs are stressing the current inter-domain routing scheme (YANNUZZI; MASIP-BRUIN; BONAVENTURE, 2005). The exponentially-growing table size on the Internet is a scalability issue that threatens the future of the Internet, and the consequences are, for example, expensive router devices and increased delays for convergence time (BENNESBY; MOTA, 2017).

On the one hand, BGP does not transport network metrics, such as capacity, bandwidth or cost, to create the topology state of its path-vector algorithm. Having those types of information is crucial to apply traffic engineering optimization tasks over network traffic. On the other hand, flooding inter-domain routing with AS internal network metrics can impose an overhead in the communication between domains, revealing sensitive operational information (for example, bandwidth utilization in inter-domain links)

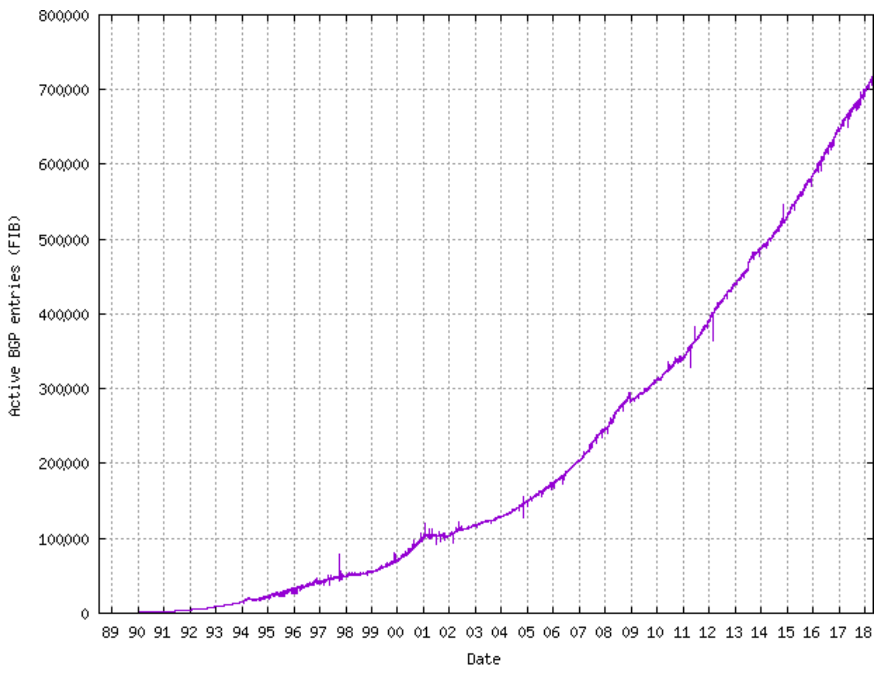


Figure 19 – The exponential growth of routing table size captured by active BGP entries at FIB since 1989.

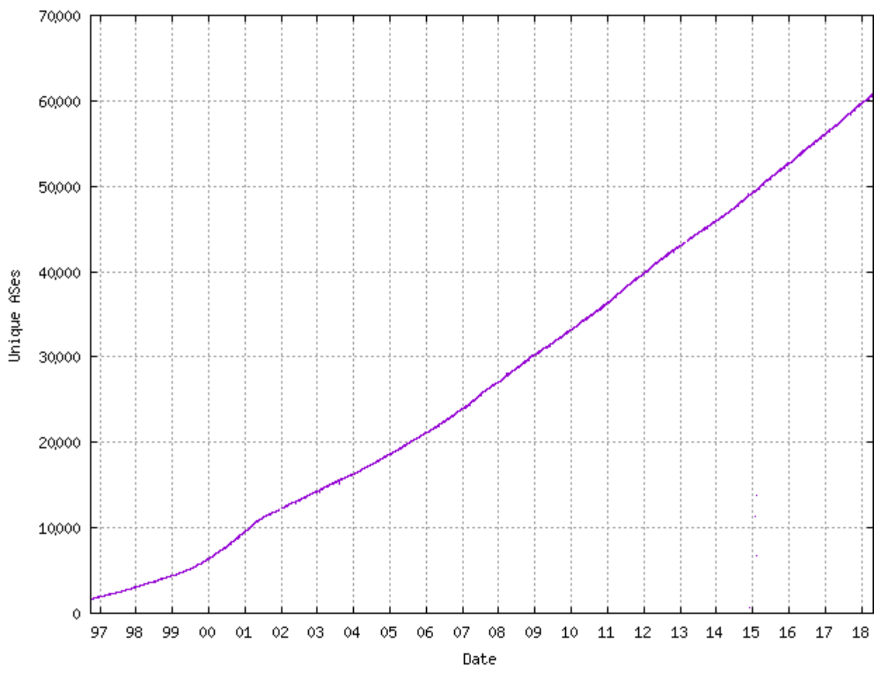


Figure 20 – The unique number of ASs on the Internet from 1997 until 2018. The data were extracted from <http://www.cidr-report.org/as2.0/>

or detailing their view of the network status. Therefore, exploring the full potential of Internet path diversity, avoiding the side-effects of TE techniques and having appropriate tools for managing traffic engineering tasks in the inter-domain environment with security and in alignment with the ASs' business requirements are the trade-offs for the next generation of inter-domain routing proposals.

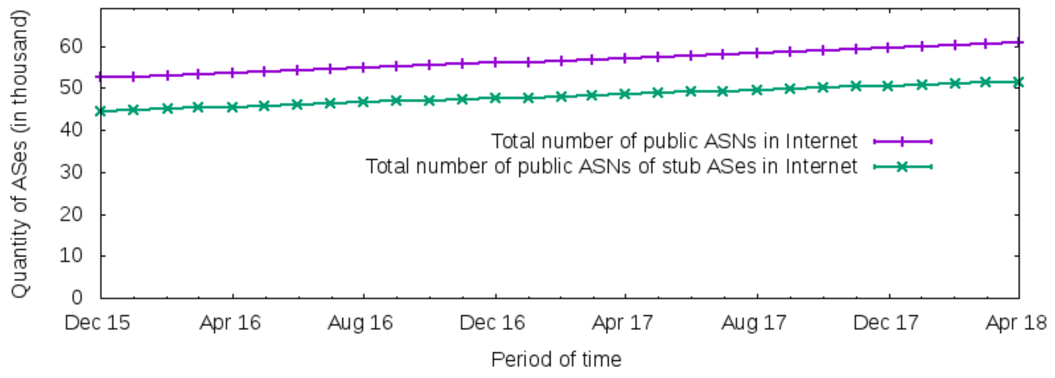


Figure 21 – The total number of ASs and stub ASs on the Internet from December of 2015 to April of 2018. Source: <http://data.caida.org/datasets/as-relationships/>

3.2 CLASSIFICATION OF EFFORTS FOR EVOLVING THE CONTROL PLANE

This section is dedicated to exploring the classification of efforts to change the inter-domain control plane. First, each one of the criteria is depicted and explained. After that, the different works to evolve the control plane of inter-domain routing are classified based on the criteria chosen.

3.2.1 The criteria

The criteria selected were *concepts*, *approach*, *control plane placement* and *explore path diversity*. Each one is depicted in this subsection.

3.2.1.1 Concepts

The concept type used in the proposals is one classification criterion for works to evolve the control plane. The concept is the set of principles and designs adopted by a work in the literature to approach the network research problem. Thereby, this survey classifies two concept types: *Traditional* and *SDN*.

- Traditional: The traditional networks are characterized as all the network logic embedded into network appliances. They have a distributed nature, and to solve a specific networking problem, they need to act individually on the affected appliances and apply manual changes in their configurations.

  Traditional network protocols have been developed and deployed by combining software and hardware into network devices (e.g., Transmission Control Protocol and Internet Protocol (TCP/IP)). In other words, closed and proprietary network systems have been produced with long development cycles (usually years) and bring new network functionalities that frequently require the acquisition of new hardware with such features. Besides, proprietary boxes often alter the configuration, as well as the compatibility of APIs (Application Programming Interfaces) across vendor (and even across different products from the same vendors (FEAMSTER; REXFORD; ZEGURA, 2014)). Therefore, legacy and discontinued products make the integration with new network devices a hard to often impossible task (SCOTT-HAYWARD; NATARAJAN; SEZER, 2015).

- SDN: SDN is emerging as a new network paradigm (NGUYEN et al., 2016; LIN; HART; KRISHNASWAMY, 2013; NENCIONI et al., 2016; SILVA; DIAS; SADOK, 2017). SDN proposes a separation between software and hardware in devices (vertical integration). Thereby, SDN has the potential to enable new technologies, network programmability and the flexibility of functionalities on network devices.

  A major feature of SDN is the decoupled control and data plane. This allows a centralization logic of the control to be fully aware of the network state, enforcing network policies, routing decisions, forwarding information, and so forth. Many technologies apply the concepts of SDN. The most notable one is the OpenFlow protocol (MCKEOWN et al., 2008). OpenFlow is an instantiation of the concepts of SDN (FEAMSTER; REXFORD; ZEGURA, 2014), which is becoming a standard *de facto* instance of SDN in the academic and industrial fields (PFAFF et al., 2012).

  Touching upon OpenFlow, it provides a programmable interface between the OpenFlow controller (the network logic) and OpenFlow Switches (forwards packets based on controller logic). Besides, OpenFlow rules follow the concept of flow, which is a sequence of packets sent from a particular source to a particular destination following a given path (KREUTZ et al., 2015). The OpenFlow rules are capable of identifying and matching TCP/IP header fields, which can provide fine-grained rules for the definition of traffic flows inside OpenFlow networks.

3.2.1.2 Approach

The approach can be a new architecture or network protocol to evolve inter-domain routing systems or both.

- Architecture: This is when a work describes a network system that details its functions and the interactions between its components or other network systems.

- Protocol: This is when a work depicts a set of rules and conventions for operation and communication between different network entities.

- Architecture/protocol: This includes the architecture or protocol approach.

3.2.1.3 Control plane placement

Other criteria adopted are the control plane placement (HELLER; SHERWOOD; MCKEOWN, 2012; BANNOUR; SOUIHI; MELLOUK, 2017), which can be as follows:

- Distributed: Each control plane element is uniquely responsible for composing the network state and performing the routing computation. Thus, those elements perform independent computation about routes and are capable of managing the portion of the network that is directly connected to it.
- Centralized: This approach is based on a single control plane that manages all the network devices.
- Logically centralized: Although multiple elements to manage the network can exist, one layer of abstraction aggregates all those elements into a seamless solution.

3.2.1.4 Explore path diversity

The BGP's destination-based forwarding paradigm indicates that all forwarding decisions about IP packets will rely exclusively on the IP destination header field. Hence, BGP will only select one “best” route per prefix when it is constructing the RIB (see Section 2.5 for more details). Thus, even when there are multiple paths to reach a given network, only one of them will be selected (HE; REXFORD, 2008; SINGH; DAS; JUKAN, 2015). Unlocking the full potential of the Internet path diversity (such as resilience or QoS) is a goal of various previous works in the literature. This work analyzes the following criteria to explore path diversity:

- Overlay: The control plane creates networks that run independently on top of another network.
- Sourcing routing: The sender has the possibility of specifying the paths that the packet will take through the network.
- Based on flows: This is the set of network rules that the control plane defines and includes the sequences of nodes that a given packet has to pass between source and destination.
- Inter-domain negotiations: The solution provides mechanisms to allow each path to negotiate to explore the path diversity of the network.
- Alternative routes: Instead of using one single best path per prefix, alternative routes try to achieve resilience, security or optimize bandwidth utilization for inter-domain interconnections exploring the availability of multiple paths.

- Not Applied (N/A): This is when the work does not provide direct evidence of how it explores the path diversity. It can occur, for example, when the research is just an architecture description or other high-level abstraction.

3.2.2 Efforts to evolve the control plane of inter-domain routing

The efforts to evolve the inter-domain routing control plane were classified as follows: *Brand new design*; *Incremental improvement*; and *Inter-domain communication.*

3.2.2.1 Brand new design

Clean-slate redesigns seem to be a very attractive approach to make a new control plane for inter-domain routing without the cumbersome backward compatibilities requirements. For traditional networks, a new protocol for inter-domain routing has the potential to overcome the limitations and drawbacks of the BGP. Table 2 compiles works with a brand new design for inter-domain routing using the criteria presented in Section 3.2.1.

Feedback-Based Routing (FBR) (ZHU; GRITTER; CHERITON, 2003) was inspired by standard engineering practice of designing dynamic systems with feedback control theory. Thus, instead of using just connectivity information for routing like the BGP, FBR creates the network state based on connectivity and structural information (such as quality and state of inter-domain links) received from routers that compose the final solution. FBR developed and used its own protocol for inter-domain routing, the Wide-area Relay Addressing Protocol (WRAP). With WRAP, FBR applies a sourcing routing scheme where the core routers of the Internet should be in charge of propagating structural information while the routing decisions occur at the routers at the edge of the network, in which they compute the network routing to achieve the end-to-end performance requirements.

Bandwidth-Aware Routing in Overlay Networks (BARON) (LEE et al., 2008) was a proposal to overcome the limitations of the default best effort Internet routing. Hence, overlay routing was used to improve end-to-end performance parameters, especially the bandwidth requirement. Furthermore, to create a feasible and scalable solution, BARON explores the Distributed Information Nodes (DIN) database that was used to distribute the node information across the network. The major drawback of overlay solutions for evolving inter-domain routing is that they normally imply additional complexity regarding the overhead that the tunnels introduce in order to be properly managed.

Fujinoki (FUJINOKI, 2008) proposed the Multi-Path BGP (MBGP) to improve the network bandwidth utilization and avoid disconnection when an external link fails. MBGP was a new network protocol to explore the multipath with a solution based on BGP updates. Furthermore, to overcome a single route for a destination network, the Multipath BGP (Van Beijnum et al., 2009) proposed a change in the BGP's path selection and path dissemination rules to explore the utilization of multiple paths in concurrence without

Table 2 – Classification for proposals to evolve the inter-domain routing with a brand new design.

Proposal and authors	**Concepts**	**Approach**	**Control Plane Placement**	**Explore Path Diversity**	**Brief description**
FBR, Zhu, Gritter and Cheriton (ZHU; GRITTER; CHERITON, 2003)	Traditional	Protocol	Distributed	Sourcing routing	Edge routers take routing decisions based on reachability and probe information of the routes that compose the routing solution.
BARON, Lee et al. (LEE et al., 2008)	Traditional	Architecture	Distributed	Overlay	Applies overlay networks to create a bandwidth-aware routing scheme.
MBGP, Fujinoki (FUJINOKI, 2008)	Traditional	Protocol	Distributed	Alternative routes	Active multipath detection for improving network bandwidth usage when link fails in inter-domain routing
Multipath BGP, Beijnum et al. (Van Beijnum et al., 2009)	Traditional	Protocol	Distributed	Alternative routes	Modifies BGP to use simultaneous multiple paths and avoid loops in routes.
AMIR, Qin et al. (QIN et al., 2012)	Traditional	Protocol	Centralized	Sourcing routing	Compute primary and alternative paths to a given destination based on topology information advertised by neighboring ASs.
NIRA, Yang et al. (YANG; CLARK; BERGER, 2007)	Traditional	Architecture and Protocol	Distributed	Sourcing routing	Allow users to choose the sequence of ISPs from their networks until reaching the final destination.
MLV, Chen et al. (CHEN et al., 2016b)	SDN	Architecture and Protocol	Centralized	Based on flows	Proposed a link vector algorithm for composing a new way of representing the inter-domain state for SDN networks.
RCS, Wang et al.(WANG et al., 2016b)	SDN	Architecture	Distributed	Inter-domain negotiations	Customer networks can explore the path diversity of SDN domains for particular applications.
SDI, Wang, et al (WANG et al., 2016a)	SDN	Architecture	Distributed	Inter-domain negotiations	Support fine-grained inter-domain routing between SDN domains.

compromising loop-freeness. Although MBGP and Multipath BGP are solutions to increase the exploration of path diversity on the Internet, the proposals were only tested in simulations, and how the network traffic should be split across multiple paths was not appropriately discussed.

Another Multipath Interdomain Routing (AMIR) (QIN et al., 2012) was a proposal for a new AS-level routing scheme to explore the use of concurrent multiple paths on the Internet. AMIR was designed to compute primary and alternative paths based on negotiating path provisioning among neighboring ASs. The AMIR scheme explores the best available paths for forwarding data in a source routing fashion. Thereby, a given domain can receive multiple paths from its inter-domain partners and choose the best one, and that has the potential to improve the experience of user applications from the network perspective.

Regarding the architecture scheme, the work of Yang et al. (YANG; CLARK; BERGER, 2007) presents a new inter-domain routing architecture called New Internet Routing Architecture (NIRA) that gives the end-host (users) and stub ASs the ability to choose the sequence of ISP that their packets can take. On the current Internet, the routes are chosen by ASs running the BGP without considering the network source to make routing decisions. The main idea of NIRA was to give the users the ability to choose routes for the Internet from a source routing approach. Thus, the users could select routes that lead to improvement of their network performance, reliability or user satisfaction. Although the NIRA's objective was to transform the Internet architecture, it requires the adoption of a new network protocol called the Topology Information Propagation Protocol (TIPP) to work properly.

Applying the SDN concepts, the Multi-dimension Link Vector (MLV) (CHEN et al., 2016b) presented a new mechanism to exchange the network view. MLV uses the OpenFlow protocol (MCKEOWN et al., 2008) in its data plane elements and enables flexible inter-domain routing in an SDN network federation. MLV is a solution based on a link vector algorithm for representing the inter-domain state of the network. With the exchanging of the link vector data structure and the fine-grained rules provided by OpenFlow, the network operator could make decisions regarding paths combining that information. Despite the fact that MLV was a proposal to evolve inter-domain routing among SDN networks, the link vector data structure imposes scalability concerns.

The Route Chaining System (RCS) (WANG et al., 2016b) follows the concept of SDN networks to allow ASs to select routes that do not follow the standard BGP algorithm and explore the possibility of diversity paths of the Internet. RCS focuses on control traffic on transit ASs from the AS source to destinations and also requires an inter-domain communication layer between ASs to enable the multiple and distributed control planes to be connected to compose the solution.

Software Defined Inter-domain (SDI) routing (WANG et al., 2016a) advances into the inter-domain support of flexible routing policies to forwarding packets, where multiple fields of the IP packet header could be used for matching (flow-level), instead of just using the BGP's destination-based forwarding paradigm. SDI also provides a mechanism to treat a large number of flow table entries required to represent the forwarding fine-grained flows. For the control plane placement, the SDI solution keeps each domain with its own control plane, which computes paths individually. To explore the path diversity between SDI domains, flow schemes are exchanged.

Summary: Clean-slate proposals are suitable for a very specific application, when used, and cannot be broadly adopted because they are incompatible with the current control plane of the Internet. In addition to traditional solutions, a new protocol proposal must be executed inside network devices or all network devices that utilize such solutions need to be modified, and such a situation may require CAPEX (CAPital EXpenditure) and OPEX (OPerational EXpenditure) to be operational. For example, the network staff would need to be trained for new proposals or protocols, as well as acquire new network devices that support those clean-slate solutions. For the inter-domain routing system with the SDN solutions, those whose approaches are not compatible with the BGP have little practical appeal for wide adoption (such as MLV (CHEN et al., 2016b)), and solutions that propagate a detailed representation of the network state suffer from scalability limitations.

3.2.2.2 Incremental improvement

Proposals made to incorporate new BGP network capabilities by changing the protocol behavior, extensions of the BGP or additional control information that is used with the BGP protocol. Table 4 presents the works related to incremental improvements of inter-domain routing.

For inter-domain routing, the authors in (FEAMSTER et al., 2004) claimed that the fully-distributed path selection computation in IP routers limits the capacity of individual ASs in terms of management scalability of path selection. Thus, they proposed Routing Control Platform (RCP), a platform that avoids the complexity of fully-distributed path computation in the inter-domain routing system by centralizing routing control logic. Due to the architecture network abstraction designed by the RCP, it did not describe how, in fact, to explore the path diversity of inter-domain routing; hence, these criteria are not applicable to the RCP. However, the RCP was a precursor of SDN concepts, in which it is indicated that the control plane of a domain should be built in a centralized way (FEAMSTER; REXFORD; ZEGURA, 2014).

In addition to centralized routing control and to enable multipath routing on the Internet (the BGP uses a single best path), Multi-path Interdomain ROuting (MIRO) (XU; REXFORD, 2006) proposed the use of overlay networks and specific packet tags to enable multiple paths for inter-domain routing. The idea was to allow a new control plane logic where ASs negotiate alternative paths, as needed, from their ISP. MIRO

Table 4 – Classification for proposals to evolve the inter-domain routing with incremental improvements.

Proposal and authors	**Concepts**	**Approach**	**Control Plane Placement**	**Explore Path Diversity**	**Brief description**
RCP, Feamster et al.(FEAMSTER et al., 2004)	Traditional	Architecture	Logical Centralized	N/A	Centralization of the network routing logic for inter-domains.
MIRO, Xu et al. (XU; REXFORD, 2006)	Traditional	Architecture and Protocol	Distributed	Inter-domain negotiations	Explore multi-paths using overlay networks and packet tags.
R-BGP, Kushman et al. (KUSHMAN et al., 2007)	Traditional	Protocol	Distributed	Alternative routes	Protect the inter-domain links against failures using pre-computed alternatives paths
YAMR, Ganichev et al. (GANICHEV et al., 2010)	Traditional	Protocol	Distributed	Alternative routes	A scheme with minimal overhead for protecting routes before the occurrence of an inter-domain link failures.
STAMP, Liao et al. (LIAO et al., 2008)	Traditional	Protocol	Distributed	Alternative routes	It executes multiple instances of the routing solution to avoid the occurrence of transient problems, such as black-holes or loops.
COIN, Silva and Sadok (SILVA; SADOK, 2017)	SDN	Architecture	Centralized	Based on flows	Override the BGP behavior in the Internet Service Providers to provide inbound traffic control.
SIREN, Kotronis et al.(KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015)	SDN	Architecture	Centralized	Based on flows	Outsourcing routing logic to Internet Service Providers.
SDX and iSDX, Feamster et al. (FEAMSTER et al., 2013; GUPTA et al., 2016)	SDN	Architecture and Protocol	Logical Centralized	Inter-domain negotiations	Override the BGP behavior in the Internet Exchanges to deploy refined routing policies.
Silva (SILVA, 2018a)	SDN	Architecture	Centralized	Based on flows	A new architecture to manage network traffic in the inter-domain using OpenFlow networks.

defaults to the single-path routing provided by the conventional BGP, but allows ASs to negotiate alternative paths as needed. This provides flexibility where required while remaining compatible with BGP. Compared to source routing, MIRO gives transit ASs more control over the flow of traffic in their networks.

An idea to protect the inter-domain links against failures is in the Resilient BGP (R-BGP) (KUSHMAN et al., 2007) proposal. R-BGP produces failover paths to mitigate the unavailability of the best path selected from the BGP decision process. Thus, when a link fails, the failover path was already computed and ready to be used, avoiding the path exploration time of discovering new paths to networks affected by the failed link. Another proposal to avoid unavailability when external links fail is the Yet Another Multipath Routing Protocol (YAMR) (GANICHEV et al., 2010). It is a resilient solution that explores advertising additional paths that are not contained in the primary path (resulting from the BGP's best path algorithm). Thus, each alternative path receives a label that identifies the links the path needs to avoid. YAMR is deployed in a distributed way, where the control plane of YAMR can be implemented as an extension of the BGP protocol.

The main contribution of the Selective Announcement Multi-Process (STAMP) protocol is executing several BGP instances inside the AS that is used to discover complementary paths (LIAO et al., 2008). The goal of those complementary paths is established paths that are not affected by the same set of network events. Thus, the STAMP requires minimal modification of the BGP process to become an operational solution and to achieve an improvement in routing stability compared with the standard BGP protocol.

The integration between SDN technologies and BGP has already been investigated and implemented by some researchers (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015; KOTRONIS et al., 2014). The integration with SDN solutions and BGP networks is vital to a practical deployment of SDN proposals. In fact, modern SDN controllers have mechanisms to exchange BGP information with BGP speaker components that behave as legacy routers. For example, the ONF ATRIUM project proposes a framework to support BGP by use of the ONOS controller (ONOS, 2017) and Quagga (QUAGGA, 2017). Another widely-used SDN controller is the Ryu (RYU, 2016) support, interworking between OpenFlow and BGP, since a BGP function is installed in the Ryu SDN framework.

The COntrol INbound Traffic (COIN) framework (SILVA; SADOK, 2017) proposed to evolve the control plane routing system with the OpenFlow protocol. The COIN framework expanded the Ryu SDN controller to provide mechanisms for controlling inbound traffic from ISP to its multi-homed ASs customers. Thus, the customers could manage how network traffic reaches them through the use of applications in the SDN controllers' ISP. Those applications, when required and allowed, override the BGP behavior of the ISP' network infrastructure to fulfil the customer's network traffic management requirements.

With the focus on merging SDN and traditional network concepts, SIREN (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015) presented a proposal to integrate the control plane of those two approaches. It combines BGP and SDN principles to improve the convergence time of the BGP at the inter-domain level. To reach that goal, SIREN allows an AS to outsource routing functionality and export it to its ISP control. SIREN is an extension of the previous idea explored in Kotronis et al. (KOTRONIS; DIMITROPOULOS; AGER, 2012). However, a pitfall of SIREN is that this approach limits the ability of the innovation of an AS, considering that a domain has to be subordinate to the requirements established by the outsourcing agreements for managing its network.

Using the idea of overriding the BGP behavior, the proposal of Software-Defined Internet Exchange (SDX) (FEAMSTER et al., 2013) and its extension Industrial-Scale Software-Defined Internet Exchange (iSDX) (GUPTA et al., 2016) explore the SDN centralized control inside the Internet Exchange Point (IXP). iSDX allows the reduction of the forwarding table size of OpenFlow devices used by the IXP participants, the creation of more flexible forwarding policies and the end-to-end enforcing of QoS. iSDX requires a brokerage system to establish multilateral peering; innovation is processed in the Layer 2 scheme (using Ethernet MAC addresses), and a third party orchestrates traffic between participants.

Silva (SILVA, 2018a) presented a new architecture to manage network traffic in the inter-domain using OpenFlow networks. The idea of the proposed architecture was to use BGP and SDN technologies to provide new mechanisms that allow different ASs to coordinate how traffic should be handled between them using network applications inside SDN controllers. Thereby, a proof of concept scenario demonstrated that stub ASs can appropriately manage network traffic towards its domain by controlling the parameters of those network applications in its ISP. The achieved results indicated the potential of the idea to apply different strategies for routing in the inter-domain environment.

Summary: Works to incrementally improve the inter-domain routing try to add capabilities to the BGP control plane, such as exploring path diversity for resilience purposes or increasing the available bandwidth of a domain. One major limitation of the BGP that is tackled by related works with incremental improvement is searching for multi-path solutions for inter-domain routing. Thus, instead of using only the one best route per prefix, the use of alternative routes has the potential to improve resilience against link failures, bandwidth availability and security (SINGH; DAS; JUKAN, 2015; CVJETIC; SMILJANIC, 2014; LI et al., 2016).

3.2.2.3 Inter-domain communication

The control communication between different domains is also an important criterion to be analyzed in related works. Table 6 depicts the related works that evolve the inter-domain control plane routing using some inter-domain communication mechanism.

For ASs that exchange control information with the BGP protocol, they can use the BGP attributes for that purpose. For example, an AS may set MED or Communities' values to alter how the control plane of other ASs selects routes (KING et al., 2016; J. Heitz et

Table 6 – Classification of proposals to evolve inter-domain communication.

Proposal and authors	**Concepts**	**Approach**	**Control Plane Placement**	**Explore Path Diversity**	**Brief description**
ADD-PATH (WALTON et al., 2016)	Traditional	Protocol	Distributed	Alternative routes	Allows BGP speakers to learn and advertise more than one path per prefix.
North-Bound Distribution of Link-State and TE Information, Gredler et al. (H. Gredler et al., 2016)	Traditional	Protocol	Distributed	Inter-domain negotiations	A mechanism to exchange link-state and TE information to external control component using a new BGP NLRI encoding format.
BGP Administrative Shutdown Communication, Snijders et al. (SNIJDERS; HEITZ; SCUDDER, 2017)	Traditional	Protocol	Distributed	N/A	It transmits a short text message describing why a BGP session was shutdown or reset.
RouteFlow, Nascimento et al (NASCIMENTO et al., 2011)	SDN	Architecture	Centralized	Based on flows	Provides virtualized IP routing services over OpenFlow enabled hardware.
WE-Bridge (LIN et al., 2014)	SDN	Protocol	Distributed	Inter-domain negotiations	Exchange control information among different SDN domains.
Inter-SDN, Bennesby et al (BENNESBY et al., 2014)	SDN	Architecture	Centralized	Based on flows	Adopted SDN applications to route packets into SDN domains.
Alto, Alimi et al. (ALIMI et al., 2014)	SDN	Protocol	Logically Centralized	Based on flows	Allows exchange network resource consumption while maintaining or improving application performance.

al., 2017). However, the BGP attributes have limited scope and effectiveness (AL-MUSAWI; BRANCH; ARMITAGE, 2016), and consequently, new signaling mechanisms between different control planes emerged to overcome those restrictions.

To tackle the ability of the BGP advertising just one best route per prefix, ADD-PATH is an extension to the BGP that uses its capability to advertise, identify and add multiple paths to a destination (WALTON et al., 2016). The standard behavior of the BGP produces and advertises only one best-path to a given address prefix. If two NLRIs are advertised with the same value of the address prefix, the latest advertisement will override the previous one in the RIB. ADD-PATH allows the advertisement of multiple paths for the same address prefix without the new paths implicitly replacing any previous ones. The main idea of this BGP extension is instead of using the address prefix as the primary identifier of a path, it uses a *Path Identifier*, in which each path receives a unique identifier, and that allows multiple values of address prefixes to exist in the RIB.

Exchanging the topology and information about the current state of the network among routing components are typically used in the interior routing protocol (for example, OSPF). Generally, the ASs are not willing to reveal their internal infrastructure or the business relationship information among other ASs because of security or business concerns. However, in some scenarios, the link-state or TE information can be shared with external components of a domain, for example, applications that require end-to-end TE (this is the case of the SIREN (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015) proposal where the external control plane was responsible for computing the routing of the customer domain or Multi-Protocol Label Switching Traffic Engineering (MPLS-TE) with Path Computation Element (PCE). The North-Bound Distribution of Link-State and TE Information (H. Gredler et al., 2016) describes a mechanism to collect and share link-state and TE information with external components by a new BGP NLRI encoding format.

Another extension for BGP to improve the communication among BGP neighbors is the BGP Administrative Shutdown Communication (SNIJDERS; HEITZ; SCUDDER, 2017), defined in RFC 8203. It tackles the lack of information when a BGP session was reset or shutdown. Thus, the solution adds a short text message as part of the notification message of the BGP (REKHTER; LI; HARES, 2006). The purpose of that message is to notify operators about the event that caused the BGP closure.

RouteFlow (NASCIMENTO et al., 2011) controls and configures flows of OpenFlow switches using Quagga (QUAGGA, 2017) as the main engine. RouteFlow becomes a proxy for the OpenFlow controller, where all the network logic follows traditional network protocols with inter-domain routing executing the BGP protocol. Furthermore, RouteFlow provides virtualized IP routing services over OpenFlow-enabled hardware, with the main idea of the proposed architecture adopting a west/eastbound interface to make a deeper integration of the routing engines of an ISP and its customers.

As recent surveys indicate, there is not yet a standard west/eastbound SDN interface to exchange network control information between different SDN controllers (KREUTZ; RAMOS; VERISSIMO, 2013; LIN et al., 2014; KREUTZ et al., 2015). Thereby, to allow the communication between SDN controllers in the inter-domain ecosystem, the authors in (LIN et al., 2014) proposed an interface for SDN to exchange reachability and topology information, the WE-Bridge (LIN et al., 2014). The goals were to present a peer-to-peer mechanism that has to be resilient, secure and exchange network control information for ASs. WE-Bridge was used by RCS (WANG et al., 2016b) and SDI (WANG et al., 2016a) to exchange control information between SDN controllers.

To exchange routing information between SDN domains (a domain that deploys SDN technologies), the Inter-SDN Routing Component (BENNESBY et al., 2014) was proposed to tackle the problem of integration between the different control planes. Using TCP connections, the new exchange mechanism is inspired by the BGP and incorporates messages similar to the BGP (OPEN, UPDATE, KEEPALIVE and NOTIFICATION), uses RIBs (RIB-In, Local RIB, RIB-Out) and simplifies the route selection decision process. The Inter-SDN claims that the way to solve inter-domain issues is through architectural abstraction (SDN applications) and extensibility properties (the network programmability).

Application-Layer Traffic Optimization (ALTO) (ALIMI et al., 2014) (defined in RFC 7285) is the most detailed specification for a standardization of the east/westbound interface for SDN controllers. It provides a network information service with the goal of exporting resources and parameters to network applications. ALTO is based on abstract maps of a network that simplify the state of a network and the applications that can effectively use them.

Summary: Traditional network approaches for inter-domain communications focus on adding new capabilities or modifications to the BGP's messages. However, those approaches require a long cycle (months or years) of development until they reach the production environment. The SDN and traditional network have one point in common: those initiatives tackle the challenge of managing a huge number of network devices that requires coordination among multiple control planes.

In fact, to overcome BGP control plane limitations for inter-domain communication, SDN approaches seek an integration with BGP and coordination between SDN controllers. Additionally, SDN network applications require complex network information that is hidden through inter-domain routing to allow the proper execution of optimization algorithms in the network (MUQADDAS et al., 2017).

3.3 LESSONS LEARNED

This section highlights the topic of the control plane for inter-domain routing and provides some insights into the content in the previous sections.

3.3.1 Inter-domain routing limitations

The BGP will never select two, or more, next hops per prefix. It always chooses a unique best hop per prefix. Different from internal routing protocols (e.g., the Equal-Cost Multi-Path Routing (ECMP)) where multiple destinations are allowed to a given prefix for load balancing proposes, the BGP standard protocol does not incorporate this behavior. Hence, for network operators to maximize the utilization of external links, they have to split the traffic using a subnet of the prefix, which has the side effect of increasing the global routing table [1].

The use of multiple links may have the consequence of improving path diversity and network reliability (CVJETIC; SMILJANIC, 2014). The BGP can filter routes by matching prefix, prefix-length and/or based on different path attributes that are associated with each BGP route and are part of an update message. However, the forwarding data plane elements can only forward the packets based on destination prefixes, once the BGP follows a destination-based forwarding paradigm (REKHTER; LI; HARES, 2006). Thereby, routers can apply their routing policies, but, they are limited to the forward decision based on the destination address of the IP packet.

Routing systems that use source addresses, policy-based routing or are based on application requirements cannot be deployed on the current Internet as this is not supported by the current BGP logic of control. Furthermore, novel network applications may require some other fields in the packet header to optimize the inter-domain routing path or simply to distinguish routing by source addresses (CHEN et al., 2016b). Those requirements are not available with the use of the BGP.

Moreover, it has long been known that the Internet architecture has several issues, despite the Internet's unparalleled success. Highly rigid and static, the traditional networking infrastructure was initially designed to operate for a particular type of traffic (text-based content), and that does not satisfy today's increasingly demanding users that request interactive and dynamic multimedia streams. Along with the multimedia trends, the recent emergence of new and innovate technologies, e.g., Internet of Things (IoT), has been pressuring the inter-domain infrastructure to support those new types of network demands.

Previous works have attempted to solve these architectural deficiencies with brand new designs or incremental improvements, which have had limited success (RAGHAVAN et al., 2012). On the one hand, the resistance to these new efforts stems from the requirement that new solutions must be mature enough and have obvious benefits. In other words, an AS will not risk its business existence in adopting technologies with unproven scalability, reliability, interoperability and consistency. Moreover, the current Internet's infrastructure took years to reach its current size, and a myriad of ASs is still expecting to reach the Return on Investment (ROI) from that.

[1] See Figure 19

On the other hand, many ASs are searching for ways to increase their profit: be more agile to address business requirement changes, provide a network that reduces cost, simplifies operations and supports new and innovative products and services. Thereby, the next generation control plane needs to be more suitable to their business needs, and it is expected that automation (DATTA et al., 2018) and intelligent network traffic control systems (FADLULLAH et al., 2017) will play an important role in evolving the inter-domain routing.

Therefore, to bridge the gap between what is expected and desirable on architectural evolvability and what is feasibly deployable has to include, among other ideas, modularity solutions with BGP compatibility and extensive support of automation.

3.3.2 New business relationships

In traditional networks, the agreements to exchange traffic are usually between ASs that have some physical connection between them, and only in specific places of the Internet is this condition not applied (for example, IXPs). Thereby, through BGP peers, ASs apply the BGP configuration that reflects the business relationships previously established and that limits the types of business relationships between ASs.

Thus, it is expected that ASs will gain more advantages of the next generation of the control plane for the Internet because this will allow them to control and explore the full potential of the Internet-wide infrastructure. For example, the observation of CAIDA's AS relationship database (CAIDA, 2016) allows the characterization of multi-homed ASs. Considering the period from December of 2015 to April of 2018, Figure 21 presents the total number of public ASN in CAIDA's AS relationship database, as well as the total number of stub ASs with public ASNs. Therefore, this graph indicates that the majority of ASs on the Internet are stub ASs, and as discussed, its control of routes is limited by the physical bound with its neighbor ASs.

Outsourcing the AS routing logic is one way to establish new business relationships on the Internet (KOTRONIS; DIMITROPOULOS; AGER, 2012). Offloading the routing functions of a customer AS to an external trusted contractor (e.g., transit AS) can optimize inter-domain traffic engineering, evolve inter-domain routing and allow the implementation of collaborative security and troubleshooting schemes.

Another idea is to transit ASs to provide mechanisms to stub ASs that manage its routes (SILVA, 2018a). A type of relationship where any ASs can establish a relationship with other non-neighbor ASs is very difficult to achieve in the current Internet architecture, since many network operational tasks (such as manual configuration) are needed. Furthermore, the relationship must make sense from the business perspective, and security mechanisms have to be deployed to guarantee an acceptable level of trust between the domains.

Thus, to establish new business relationships between multiple ASs, a reliable and secure mechanism has to emerge for inter-domain routing. The manual configuration of the BGP and the use of TCP (KEVIN et al., 2010) to exchange NLRI are not enough for the next generation of the control plane for inter-domain routing that has to incorporate new mechanisms to exchange and establish network relationships through any AS on the Internet to enable dynamic resources, bandwidth and routing.

Previous works in the field indicate the values of collaboration among ASs with different business models (CHANDA; WESTPHAL, 2013; WICHTLHUBER; REINECKE; HAUSHEER, 2015; RAO et al., 2011). For example, exploring a more intensified collaboration between ASs that produces (e.g., Content Delivery Network (CDN)) and consumes the content (e.g., Access Networks) can produce gains for all participants (POESE et al., 2010; POESE et al., 2012). Moreover, with new inter-domain communication mechanisms, it will be possible to deploy visionary solutions such as an economy plane for the Internet (WOLF et al., 2014).

3.3.3 SDN as an enabling technology

One of the major features of the BGP is the scalability of the protocol due to its fully-distributed nature. The size-effect of distributed control of the BGP is causing the protocol "chattiness" (YANNUZZI; MASIP-BRUIN; BONAVENTURE, 2005) and the long convergence time (BENNESBY; MOTA, 2017), since every BGP speaker has to generate the global view of the network state to make decisions about routing. The centralization of inter-domain routing control can be used to mitigate the BGP issues (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015; KOTRONIS, 2015; THAI; De Oliveira, 2013). A bird's eye view of the centralized solution has the potential to identify misconfiguration, avoid route leaks, address the satisfaction and dissatisfaction among ASs (CARDONA et al., 2016) and understand and perform troubleshooting; all sorts of undesired network behavior will be exposed to the centralized control.

Previous proposals that use exclusively traditional network technologies (YANG; CLARK; BERGER, 2007; FEAMSTER et al., 2004) required extensions for the BGP or a brand new network protocol to make them operational. This results in a difficult scenario that is not easy to deploy or manage for ASs because of CAPEX, OPEX, security and reliability concerns. With SDN technologies, the consistency of the network state is encouraged by the principle of logical centralization (BANNOUR; SOUIHI; MELLOUK, 2017). The inter-domain routing centralization has the potential to benefit the application of traffic engineering techniques, enforcement of routing policies, network troubleshooting and other desirable features. For example, the proposal of iSDX (GUPTA et al., 2016) uses the SDN centralized control in an IXP to allow the creation of more flexible forwarding policies and the end-to-end enforcing of QoS. Thus, the adoption of SDN technologies can solve the management decentralization problem of the traditional network, as discussed in Chen et al. (CHEN et al., 2016a).

Furthermore, new network services can emerge if the current inter-domain routing system allows the evolution of the routing control plane. The traditional network mostly applies the decentralization of the routing decision, and with the advent of SDN concepts, the logical centralization of the routing control plane may enable inter-domain routing evolution. Thereby, a good question to pose regarding SDN is the following: Why have SDN technologies not yet changed the inter-domain landscape? First of all, it is worth noting that the SDN paradigm is a brand new approach to design networks (although it inherits ideas from other approaches (FEAMSTER; REXFORD; ZEGURA, 2014)), which is different from the traditional network that has been the main paradigm to construct networks since the Internet's inception. Therefore, the technologies that instantiate SDN concepts are in their early stages, as they are not mature technologies, and wide adoption is not expected for the next couple of years.

The most successful SDN technology is the OpenFlow protocol. This protocol has some hindrance regarding memory and signaling utilization (SILVA, 2017) because OpenFlow switches usually use TCAM as the main memory to match a flow. That type of memory is very fast, however having the drawback of being expensive with a high energy consumption. The fine-grained OpenFlow rules can represent a cumbersome inter-domain environment where the number of prefixes (or flows) is tremendously high (WANG et al., 2016a). Flow management is almost mandatory for the real deployment of the OpenFlow protocol in inter-domain routing, and that problem can be understood as an extension of the rule placement problem inside OpenFlow networks. A detailed survey about flow management can be seen in the work of Nguyen et al. (NGUYEN et al., 2016).

Another drawback of the OpenFlow networks is the reliability between controllers and the data plane elements (OpenFlow switches). Once the control and data plane is decoupled, when the control becomes unavailable, the forwarding elements may not continue to work appropriately. Some works claim that the data plane elements need to have some capabilities to fix those scenarios, and a mix of control and data plane programmability can be the key for future network solutions. Ideas for allowing data plane elements to become more flexible and programmable are in their infancy, such as P4 (BOSSHART et al., 2014) or OpenState (BIANCHI et al., 2014), and those technologies could resolve the flexibility and performance of OpenFlow switches.

Therefore, to be practical, an OpenFlow solution must manage the flows (installation, aggregation and eviction) and maintain a trade-off among rule installation and the signaling overhead between OpenFlow switches and the OpenFlow controller to maintain the availability of the network (NGUYEN et al., 2016; SILVA, 2018b). For ASs, and especially ISP, new proposals cannot be adopted for inter-domain routing that represent a risk (reliability, availability, performance, security and other requirements) to their business unless substantial profit is imminent (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015).

3.4 CONCLUDING REMARKS

There is still much to be done until the BGP control plane is fully replaced as the main routing core system of the Internet. None of the efforts researched in this work provide all the features to reach that goal. SDN technologies seem to be a prominent technical field for new network architectures, and they are receiving much academic and industrial attention. Beyond the technical requirements, the new proposals to evolve inter-domain routing must be aligned with network operator business vision.

4 THE ARCHITECTURE PROPOSAL

The main idea of this architecture is to exploit the global view of the network provided by the SDN and exchange control information between different ASs to manage how inter-domain traffic should be performed. The Figure 22 depicts the high level view of the architecture context. Once the AS has SDN's capabilities, it can exchange control information with its clients to discover what is the best way to deliver network traffic. Thus, instead of just relying on BGP mechanisms to execute traffic engineering tasks, a new control channel can be used for that purpose. However, to an SDN be able to provide new capabilities and to create a new logic inside the SDN domain, the network controller must be capable of enabling new types of functionalities.

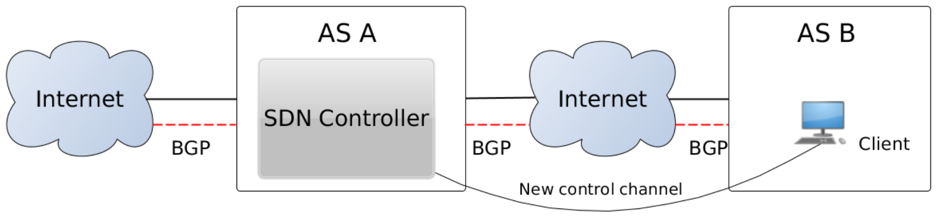


Figure 22 – The context of the architecture.

Thus, this Chapter is dedicated to presenting the architecture to control traffic in inter-domain routing using SDN technologies. An overview of the controller that enables the architecture functionalities is presented in Section 4.1, where a description of each component is done. Section 4.2 tackles the challenges of manage flows inside an SDN network. After, in Section 4.3, the dynamic of the proposed architecture is presented, depicting how the architecture establishes a peering session between domains, how it can explore the multipaths in the inter-domain environment and avoid creating loops during the path diversity exploration. The Section 4.4 is dedicated to the description of OpenFlow applications to manage network traffic using the proposed architecture. The last section, Section 4.5, provides the concluding remarks of the chapter.

4.1 THE CONTROLLER FOR AN ARCHITECTURE TO EVOLVE INTER-DOMAIN ROUTING

The architecture to evolve inter-domain routing requires to use of SDN technologies to export new capabilities not currently available in BGP. Hence, this thesis developed an SDN controller called **CO**ntr**OL** Traffic (COOL)[1]. The overview of the SDN controller is depicted in Figure 23, with the three planes of the SDN architecture. The presentation of the controller will following the bottom-up approach.

[1] Publicly available in: <https://github.com/walberjose/COOL>

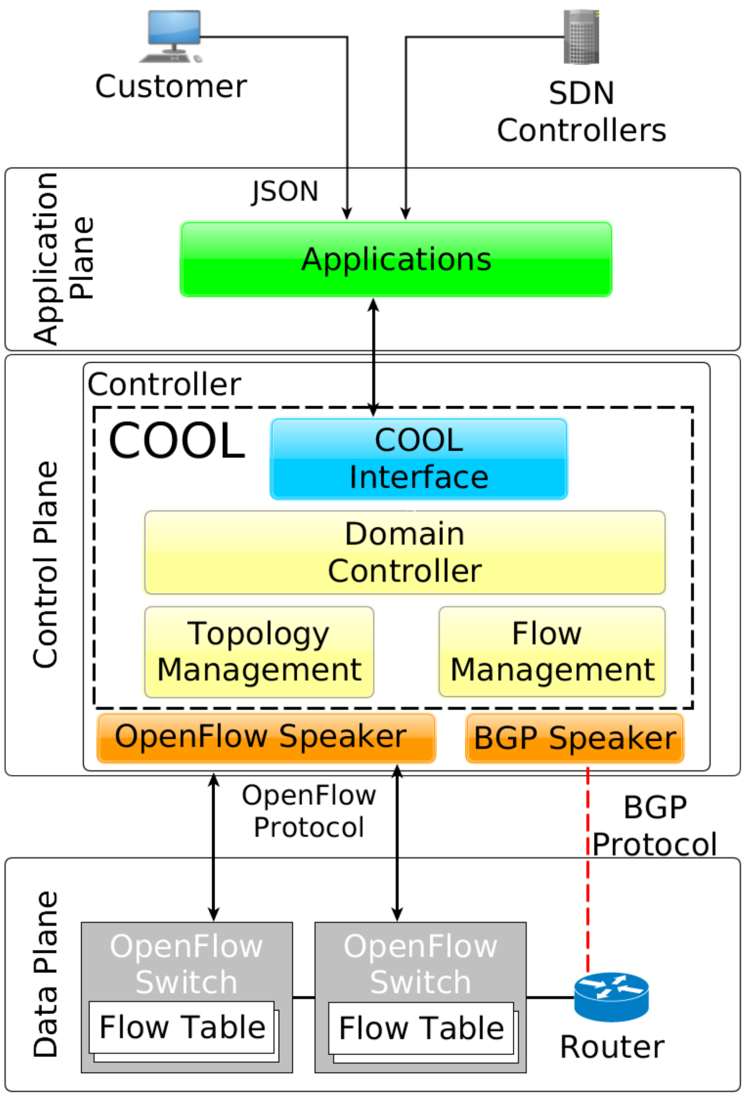


Figure 23 – The proposed architecture described by the application, control, and data planes of the SDN architecture.

In the *Data Plane*, the *OpenFlow Switches* and *BGP Routers* connect to the *Control Plane* using OpenFlow and BGP Protocols. Thus, the controller has the OpenFlow protocol capabilities and a modified *BGP Speaker* that provides access to all *Adj-RIB-In* (see Section 2.5) between all BGP neighbors.

Because the BGP selection process selects only one best next hop per prefix, some changes are required in the *BGP Speaker* component to allow extract maximum information from the BGP Peers (details in Figure 24). Once with the routing information learned from the neighbors, COOL can allow new types of SDN application to take decision about how to select paths. And after the decision made by the SDN application, the paths are installed in the Data Plane using OpenFlow rules by the *OpenFlow Speaker* component.

Figure 24 indicates the flow of information between the BGP Speaker component and the COOL controller. First, all BGP peers establish a connection to the *BGP Speaker*

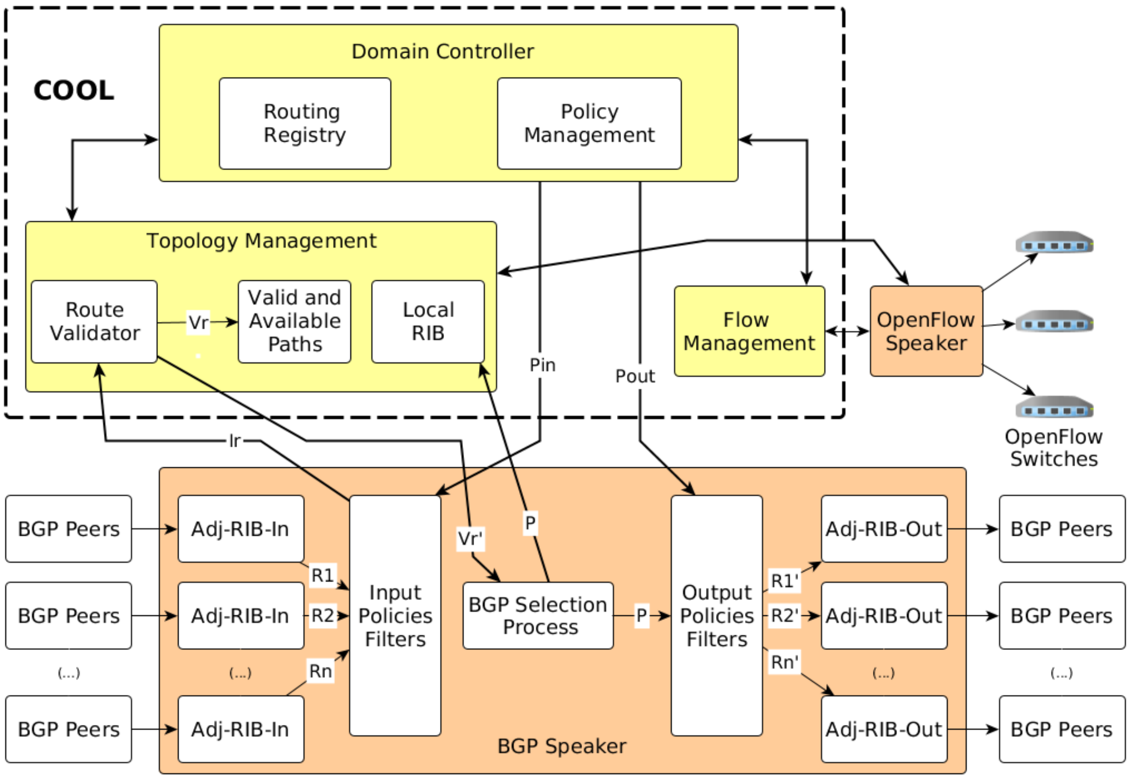


Figure 24 – The flow of information inside the BGP Speaker component.

component. Those peers can be internal or external. Let us assume that all those peers are enumerated with a number. Thus, the routes learned from each peer advertisement can be presented by R_i, where i is a unique identifier for the peer. Denoting P_{IN} an abstraction of the input policies of the domain that deploys the proposed architecture, the set of received input routes is represented by:

$$I_R = \sum_{1}^{n} R_i - P_{IN}$$

Then the *Topology Management* component exclude all invalid routes, including fault AS path or fault prefixes, using the *Routes Validator* component. As the result, the set of valid routes is produced, V_r'. Considering that the BGP selection process is represented by the function B_P, then V_r' is used as the input for B_P. The result is $P = B_P(V_r')$, where P is a set of all primary routes (best ones). Thus P composes the *Local RIB* database of the domain. To advertise the output routes, R_n', for others nth BGP Peers, those peers receive the outcome of $R_i = P - P_{OUT}$, where P_{OUT} is the output policies defined for each peer.

The COOL uses the global network view and extracts as much routing information as possible about prefixes advertised from its peers. However, to allow new network capabilities, it is natural to extend the SDN controller. Thereby, COOL has its components for embracing the proposed architecture: *Topology Management*, *Flow Management*, *Domain Controller* and *COOL Interface*.

4.1.1 Topology Management

This component is responsible for acquiring physical network information, such as link and switch identifications and features (e.g., supported link rates at each switch port, etc.). It also registers all switches that enter and leave the SDN network. If the architecture uses the OpenFlow protocol for the SDN network, then all *OpenFlow Switches Enter* and *OpenFlow Switches Leave* events are notified by the controller. Those type of events provides a list of switches in the network, as well as the links between them. Thereby, the *Topology Management* component can create a digraph[2] data structure representing the underlaying physical infrastructure.

To ensure backward compatibility with BGP, *Topology Management* module is also capable of acquiring NLRI information (AS-Path length, network prefix, etc.) by accessing information in the *BGP Speaker* component. Therefore, both local and external routing information about prefixes reachability is available and then them are combined to build a global topology state of the network.

When using OpenFlow, *Topology Management* component also receives information about the states of the OpenFlow table (information inside the OpenFlow switches). Thus, the *Topology Management* module monitors the underlying infrastructure to maintain its topology consistent with network events. When a link suffers a failure, it is expected that the *Controller* receives information on its new state from the OpenFlow switch via OFPT_PORT_STATUS message notification (PFAFF et al., 2012). Any topology change must be communicated to the *Flow Management* module for processing.

To improve the information about the topology of the network inside the *Topology Management* component, others protocols (different of OpenFlow protocol) can be enable by the controller. For example, to create a logical abstraction of the representation of the physical topology a discovery protocol can be instantiate to make the representation of the topology always updated, for example, the Link Layer Discovery Protocol (LLDP), or for fast recovery from link failures, Bidirectional Forwarding Detection (BFD) can be enabled and used (see Subsection 2.3.1 for more details).

It is notable that the network administrator of the domain can include some manual configuration that describes the hosts and networks. This is useful for connecting the BGP peers inside and outside of the domain. Traditionally, the BGP configuration is executed manually, and this component allows this type of procedure.

4.1.2 Flow Management

To allow network traffic from one AS to another, it is required two types of routing system: intra-domain routing, which is the routing process inside the single autonomous system; and inter-domain routing, which is the process of routing traffic between ASs. Because

[2] A digraph, or directed graph, is a graph that made up of a set of vertices connected by edges, where the edges have a direction associated with them.

COOL uses BGP to manage reachability of external networks, and it is based on flows, COOL requires a scheme to allow packets to reach network borders.

In an OpenFlow network, the controller is responsible for creating new flows into the network (SILVA, 2017). Thus, COOL classifies two types of OpenFlow switches: border; and internal. A *border* is an OpenFlow switch that has at least one link connection to another AS, and an *internal* is a switch that all links are connected to *internal* or *border* switches.

Thus, in COOL, a flow is defined as a tuple: $(IP\_src, Mask\_src, IP\_dst, Mask\_dst)$, in which IP_src represents the IP prefix of a network source with its network mask $Mask\_src$, and IP_dst is the IP prefix of destination with its network mask $Mask\_dst$. That definition of flow is used to create flows between *internal* switches. Thus, a sequence of packets sent from a particular source to a specific destination will following a given path, where the packets with the same the IP header fields belongs to a flow.

For a connection between *border* switches, a tunnel is established. Whether an external link suffers a failure, all the traffic through that link will be redirected. COOL uses a *tag* to identify a connection between any two *border* switches, with the requirement of the value of a *tag* must to be unique. The main idea is to avoid the naive approach where all the OpenFlow switches in a flow have the tuple installed, and the use of a *tag* mitigates the memory consumption of the network (see Subsection 4.4.2 for more details).

Because BGP policies have to reflect business requirements of an AS, *Flow Management* is subject to the routing domain policies passed by the *Domain Controller* module to avoid routing policies conflicts. Besides, the *Flow Management* component computes the paths to be installed into the OpenFlow network of the domain that uses the proposed architecture. It sends instructions to the *OpenFlow Speaker* component to install the OpenFlow rules in the OpenFlow switches.

Flow Management also uses information about network topology and hosts (stored and available by *Topology Management*) to compute flows inside the OpenFlow network. The Section 4.2 is dedicated to explain how this computation effort is performed.

4.1.3 Domain Controller

This component is responsible for applying or repealing the policies of the domain that deploys the proposed architecture. Thus, if an *AS app* needs to explore multiple AS paths for a given prefix, the *Domain Controller* will allow, or not, the use of multiple next hops for that prefix.

To reach that goal, *Domain Controller* has two modules: the *Policy Management* that is responsible for avoiding two (or more) *AS app*s creating rules that go against each other or the AS policies; and *Routing Registry* is responsible for store and query Routing Registry (RR) information about routes.

Furthermore, *Policy Management* can verify if routing decisions based on the policies provided from the domain are valid or not. This avoids two (or more) *AS app*s making conflicted decisions about the routing, each *AS app* is associated with a customer AS, and each customer has their own prefixes registered inside the *Routing Registry*. Thereby, verifying whether a given prefix belongs to a customer AS is conducted through querying the *Routing Registry* database and, with that information, it is possible to avoid, or discover, if the routing objective of one *AS app* negatively affecting the routing objective of the other.

The SDN data plane technologies (e.g. OpenFlow protocol) provide various fields for matching packets including different TCP/IP protocols (SILVA, 2018b). The proposed architecture can explore the capabilities provided by the data plane technology, and it defines the routing policies based on the 4-tuple: $\{P_{src}, ASN_{src}, P_{dst}, ASN_{dst}\}$, where P_{src} and P_{dst} represent the source and destination prefixes, respectively, and ASN_{src} and ASN_{dst} the ASN for the source and destination of a given domain, respectively. Therefore, the 4-tuple can provide new types of fine-grain routing policies for inter-domain routing with different granularity ($*$ is a wildcard symbol):

- Policy-based on destination ASN: the pattern for matching this type of policy can be reachable using the tuple $\{*, *, *, ASN_{dst}\}$. Thus, all network traffic that matches with this rule will be redirected to the ASN_{dst} domain;

- Policy-based on destination prefix: $\{*, *, P_{dst}, ASN_{dst}\}$. It is assumed that the prefix P_{dst} must belong to the ASN_{dst} and each P_{dst} is advertised by only one AS to this policy become valid. Note that this type of policy is equivalent to the application of BGP routing policies. To keep the consistency and backward compatibility with BGP, the "best" route for prefix P_{dst} (assigned by the BGP Decision Process of *BGP Speaker* component) of BGP is used as the default behavior of the proposed mechanism and must produce the lowest priority rules to guarantee that more specific rules will be select (whether exist);

- Policy-based on source ASN: $\{*, ASN_{src}, *, ASN_{dst}\}$. With this policy, network traffic coming from different source ASs can be required to be routed differently. This new type of routing policy offers a better granularity than routing policies of BGP (destination-based);

- Policy-based on source prefix: $\{P'_{src}, ASN_{src}, *, ASN_{dst}\}$. This type of policy is, in fact, a specific case of the policy based on source ASN, where the prefix P' is explicit informed;

The set of actions that a given AS can take using in the proposed architecture are:

- Drop: discards packets that matches the routing policies;

- Forward: accept the packets and forward to one of the AS neighbors.

Implementing the new types of routing policies requires a data plane with new capabilities and, hence, forward traffic based on other TCP/IP fields, instead of just the IP destination value. When a domain that uses the OpenFlow protocol (MCKEOWN et al., 2008), which is an SDN technology, or other network programmable appliances for the data plane elements, it can enable the potential to evolve the network logic.

4.1.4 COOL Interface

The *COOL Interface* is used to overcome the BGP limitations regarding control messages. New types of control messages can be created and to be exchanged between different domains. Besides, this component allows communication between the ASs, others controllers and exchanges control information about prefixes to customers of the domain. It helps the *Domain Controller* to perform some control tasks as well.

The communication should use a secure channel (e.g., Hyper Text Transfer Protocol Secure (HTTPS)) to send and receive messages in readable format (e.g. JavaScript Object Notation (JSON)). And it should also be responsible for exchanging Authentication, Authorization, and Accounting (AAA) messages to Customer AS. Although the security requirements should play an important role in a production deployment of the proposed architecture, the focus of this thesis is on functionalities that SDN can bring to a new architecture for inter-domain routing, and not develop a product, or a complete security solution for the proposed architecture. Therefore, the few messages of the *COOL Interface* are:

- Get RIB: receives information about the valid and available prefixes;
- Set Multipath: informs the paths that the packets should take for an specific application of *Customer AS*;
- Get Flow: receives statistics about the installed flows in the data plane. It also provides information to the *Applications* regarding multiple paths and the bandwidth utilization for a given prefix, and verifies the quantity of network traffic destined for the prefixes of the *Customer AS*;

Others messages are used for the execution of new routing application. Thereby, the *COOL Interface* provides information to *Applications* about the routes. For example, if a change in the RIB occurs (e.g., a withdraw prefix network) the *Domain Controller* component notifies the *Applications* through *COOL Interface*, and then, the applications can update to the new the network state.

4.1.5 Applications

The *Applications* is the place where the new logic for the inter-domain routing can occur and, thus, the control of traffic is executed inside the application plane of the proposed architecture. A customer AS initializes the *AS app* to manage the network prefixes of the AS that deploys the proposed architecture with the goal of instructing how traffic will be handled. The *AS app* also receives the network status from the proposed architecture. For example, the bandwidth utilization of each prefix owned by the *Customer AS*, or the available paths to reach its networks. With that information, the network operation of the *Customer AS* can send control information to configure how much traffic each path will receive for each prefix in use.

The *AS app* is responsible for exploring the multiple paths available for a given prefix if it is required. The information about those paths come from the *BGP Speaker* component that learned all the prefixes advertised by the AS neighbors through external BGP peering. Furthermore, with the information about multiple paths available, the *Customer AS* can indicate how the traffic will be treated and distributed to those paths.

The BGP protocol uses just one "best" next hop for each prefix even if other paths to those prefixes are valid and available. BGP is not a multipath protocol, and one good reason for that is that exploring multiple paths on the Internet can affect the performance of Transmission Control Protocol (TCP) connections, the most used transport level protocol in the Internet (AGER et al., 2012).

Thus, distributing packets belonging to a connection through multiple paths can result in poor performance of high-level protocols of the TCP/IP stack (LIU et al., 2014b) (e.g., reordering packets of TCP (SINGH; DAS; JUKAN, 2015)). In particular, it is often the case on the Internet, which connects heterogeneous types of network appliances and protocols, that the network suffers from different delays among paths. To avoid the problem of exploring path diversity on the Internet, the proposed architecture guarantees that the association of the source and destination of the IP packets will be forwarded through the same path. Then, a 2-tuple composed of the IP source prefix and the IP destination prefix is adopted to define a flow in the OpenFlow network. Therefore, for a given TCP connection, it will use the same path during its time life (if the path is valid and available).

Aligned with the 2-tuple flow definition, the *AS app* of a customer AS can also check the health of each path in use by sending control messages to multiple paths. Thus, if an inter-domain path suffers a problem (such as link failure) or a transient loop occurs, the *AS app* will be notified not to use that problematic path.

4.2 MANAGING FLOWS

Traditional networks evolved through embedding software into the network hardware in the form of protocols that work in a distributed matter. With SDN, the controller is responsible to manage and control the flows and to keep the network available in the presence of failures.

To fulfill this challenge, it was developed, in this work, an engine for manage flows inside the SDN network. The piece of software responsible for compute flows is the *Flow Management* component of COOL controller. COOL controller also executes a scheme to avoid inconsistency in flow creation that mitigates the problem of flow creation for stateful SDN applications, and the **HY**bri**D** **R**esilience Mech**A**nism (HYDRA)'s algorithm, which is a new resilience mechanism for SDN network.

4.2.1 A scheme to avoid inconsistency in flow creation

This subsection provides one solution for the problem of multiple *Packet-In* events that can cause inconsistency in the network view of OpenFlow applications. Thus, as previously presented (as shown in Subsection 2.3.3), the OpenFlow stateful applications (an application that stores the network state) require to receive the consistency of network changes. Multiple *Packet-In* packets sent to the controller can alter the logic behind stateful applications because a typical OpenFlow application relies, for example, on *Packet-In* events to create new network flows. Therefore, this Subsection proposes a scheme for avoiding inconsistency in OpenFlow stateful applications caused by multiple flow requests.

The conceptual scheme for mitigation of multiple *Packet-In* events is depicted in Figure 25, and it required the network operator to define the *Definition of flow* to be used in the OpenFlow network. Those flows can be composed of actions (e.g. forward, drop, and modify) and a match pattern that can use any subset of a packet's header field of the protocols supported by the OpenFlow specification (ONF, 2017).

The *Definition of flow* allows the controller to create flows in advance (*Proactive* flow creation (SILVA, 2017)) into the switches. Thus, before packets matching those rules arrive in the network, the switches will already have information about how to handle the incoming traffic.

After the flow definitions, it is required that the *Priority of flow* is given. For example, the network operator can define that IP packets carrying TCP protocol should be more important than IP packets carrying User Datagram Protocol (UDP) (or the other way around). The goal is to prioritize some flows in order to speed up the OpenFlow table lookups, once the priority of flows is the first criteria to be observed during table lookups in OpenFlow. Thus, *Priority of flow* has to indicate those flows that will receive priority, as well as rate limits if it is applicable.

Also, all flows defined by the network operator is tightly adjusted by the "max_len" value, whose goal is to avoid waste of bandwidth between switches and controller. This adjustment uses *Size of flows* present in Table 8, which shows that the sum of header fields size for composing a flow is used to set the "max_len" value of rules that generates a *Packet-In*. Thereby, only the quantity of bytes indicated in the size of "max_len" for

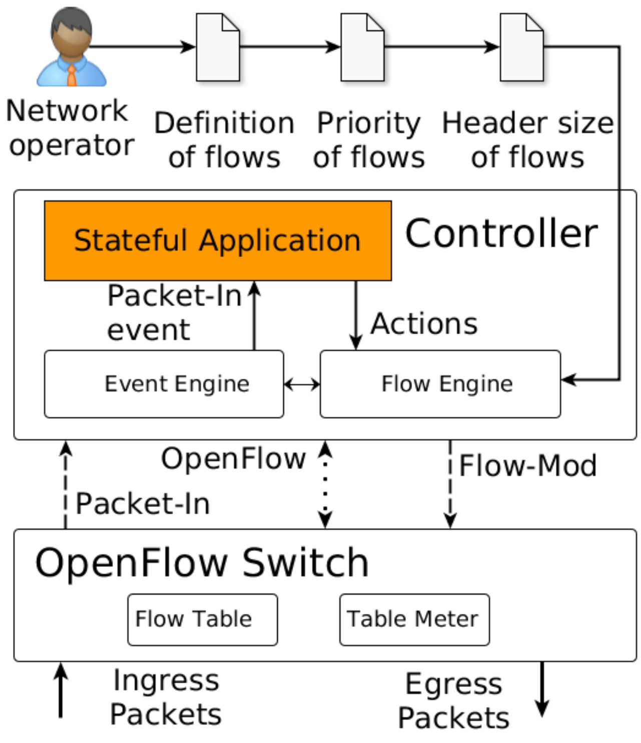


Figure 25 – Conceptual scheme for mitigation of multiple *Packet-In* events.

Table 8 – Size of “max_len” for each definition of flow.

Header fields	Value of “max_len”	Definition of Flow
Ethernet	14 bytes	(In_port,Ethdst)
Ethernet+ARP	42 bytes	(In_port,ARPdst)
Ethernet+IP+ICMP	50 bytes	(IPsrc,IPdst,IPproto)
Ethernet+IP+TCP	54 bytes	(IPsrc,IPdst,IPprot,TCPsrc,TCPdst)
Ethernet+IP+UDP	42 bytes	(IPsrc,IPdst,IPprot,UDPsrc,UDPdst)

each definition of flow is used in a *Packet-In* packet, ignoring the other bytes belonging to the incoming packet.

All information about flows are sent to the *Flow Engine* that populate the flow tables of the *OpenFlow Switch* through *Flow-Mod* messages of the OpenFlow protocol. For example, if the flow received a low priority and a rate limit, the controller will instruct the switch to create an entry in the *Table Meter* for that flow. The *Flow Engine* is also responsible for storing the flows created and active in the switches.

Furthermore, in the controller, it is required a modification of the *Event Engine* (see Figure 25), which belongs to the controller framework and manages the *Packet-In* events. Algorithm 1 presents the modification of *Event Engine*. The algorithm requires the topology information of the network, represented in a digraph $G(V, E)$, where V are the

Algorithm 1: Algorithm of Event Engine

```
Require: $G(V, E)$, $flows$ and *opi*
Ensure : Only one event generates a change in state of a stateful application, even
         in the presence of multiple *Packet-In*
1 if *get_flow(opi)* ***NOT in*** $flows$ then
2     send_to_applications(*opi*) ;
3 else
4     *actions* ← get_actions(*opi*,*G*,$flows$);
5     apply_actions(*actions*,*opi*) ;
```

switches, and E represents the set of links between each switch. It also requires that all the flows already created are stored in the data structure $flows$, and the representation of *Packet-In* is *opi*. In Line 1, it checks if the *opi* is not already in $flows$. If this is the case, then the *opi* is a brand new solicitation and needs to be treated for an application. Otherwise, the algorithm performs the actions that the application defined for that flow (lines 4-5). In line 2, the *Event Engine* sends the event to the applications that require to receive the *Packet-In* packet in a controlled way. The final consideration is that the Algorithm 1 must be deployed in a non-preemptive code inside the controller, to avoid inconsistency in the manipulation of $flows$ data structure.

4.2.2 A hybrid resilience mechanism for OpenFlow networks

This section presents the algorithms for the HYDRA that is a hybrid resilience mechanism used for creating flows inside OpenFlow networks for the COOL controller. This resilience mechanism is used for the controller to manage flows inside the SDN network. Thus, HYDRA has two phases. In the first phase, a given path is protected by the additional OpenFlow rules with OpenFlow Fast Failover Group Table (FF) (see Section 2.2.2 for more information about the FF technique). When a link (or a set of links) fails, the second phase kicks in and restores the consistency state of the network with the goal of optimizing all paths affected by the link failure. The next subsection presents an example that explains and clarifies the phases of this hybrid resilience mechanism.

4.2.2.1 An example

An example to applying the HYDRA is in the simple topology depicted in Figure 26, in which the source hosts (S1, S2 and S3) communicate with each destination host (D1, D2 and D3). In Figure 26a, the path [A, B, C] is the primary path for flows from source to destination hosts. The first phase of the resilience mechanism applies a protection rule from Node B to Node C (path [B, A, D, E, F]). Thus, if any of the links A-B or B-C suffers a failure, the backup path will be used automatically with the OpenFlow feature of FF. That way, path [A, B, C] is protected.

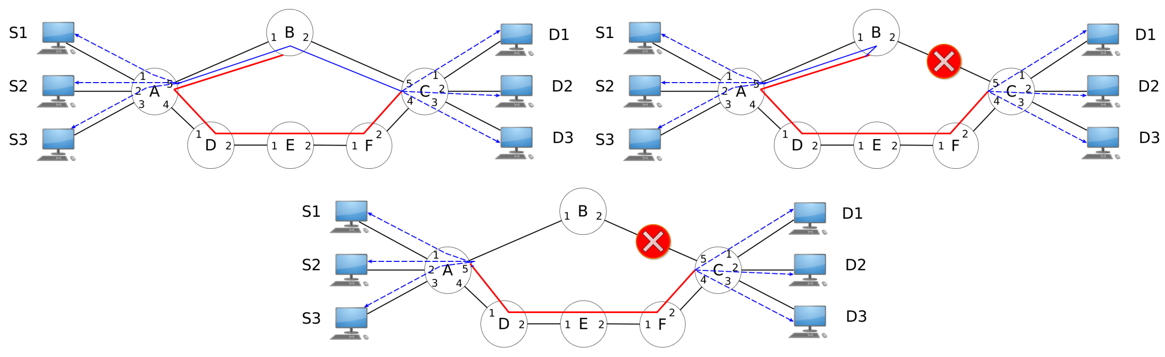


Figure 26 – HYDRA dynamics before and after a link failure. (**a**) HYDRA before the link B-C failure. Path [A, B, C] is protected. (**b**) HYDRA after the link B-C failure (proactive protection). (**c**) HYDRA after the protection and restoration phases are applied.

If a link that belongs to the primary path suffers a failure, the protected path will be used, as indicated by Figure 26b, in which the link B-C is no longer available. As a consequence, the packets will travel through a backup path until reaching Node C (destination node). Thereby, even if the link A-B fails, all the flows from source hosts will continue to experience reachability to destination hosts because the backup path also protects that link. This is the basic idea of HYDRA's protection phase, where the main goal is to make the paths highly available.

Nonetheless, the use of a backup path can result in a non-optimal network utilization because it is possible that the flows will travel a longer distance than necessary after a link failure. A restoration phase is fundamental to make all the paths optimal again. Often, the shortest path algorithm is used to create optimal paths (in which the path length is the metric), but an optimal path can be defined with any metric that a network operator stipulates.

A resilience mechanism deployed in the network hardware, such as FF, can lead to an inconsistent global view of the network because when a link failure occurs, all multiplex actions of FF are performed inside the OpenFlow switch without the consent of the controller (SILVA, 2018b). In this situation, the controller must recover the current network state.

Different from previous works that adopted specific protocols for discovering a link failure (e.g., Bidirectional Forwarding Detection (BFD) (D. Katz and D. Ward, 2010b)), HYDRA uses the OpenFlow messages OFPT_PORT_STATUS sent by OpenFlow switches to the controller. Those messages contain the identification of the switch and port affected by the failure. Therefore, the controller acquires information necessary to recover the network state consistency and perform corrections to non-optimized flows.

Once the controller is aware of the a link failure, it starts a restoration phase to recover the consistency of the network state and makes flows that are using backup paths optimal again. Figure 26 presents the scheme for the restoration of the path between Nodes A and C, after the link B-C failure. That concludes HYDRA's final phase.

4.2.2.2 Notation and concepts

For brevity, first, the notation used is summarized in Table 9, and an individual description for each item is provided. The notation is used to compose the HYDRA algorithms, and those are detailed in the next Subsections.

Table 9 – Notation adopted.

Notation	**Description**
$G(V, E)$	The network topology, where V denotes the set of nodes (switches) and E the set of edges (links between switches)
O	Set of endpoints (source and destination of a flow)
I	A set of path identifiers
s	Source endpoint (source a flow)
d	Destination endpoint (destination a flow)
$get\_port(v, d)$	Function that returns the output port for destination d of switch v
$MT(\bullet, \bullet)$	Function that returns the match for a given argument
$ofps$	OpenFlow packet port status event
opo	OpenFlow packet output event
$send(\bullet, \bullet)$	Function that sends an *OFPFlowMod* packet.
F	All flows installed in the network.
v	An OpenFlow switch
l	A failed link
$\|\| \bullet \|\|$	Provides the number of elements in a given set

Thus, before delving into the algorithms, some concepts need to be defined. One of them is the concept of endpoints. These are places inside an OpenFlow network where the network operator has some interest for packets to reach that location (notation O; see Table 9). For example, an endpoint can be a host, a switch port connected to a load balancer appliance, the ingress/egress of the backbone network, and so forth. Thus, an endpoint consists of two elements. The first one is the source, where the packets are identified as belonging a specific flow, and the destination, the place where the packets are released from the flow. Furthermore, the path connecting the source to the destination is composed using a tunnel.

This work models an OpenFlow network as a directed graph (or digraph) $G(V, E)$, where V is a set of nodes (e.g., OpenFlow switches) and E a set of edges (e.g., network links). For the creation of the network state, the OpenFlow controller usually uses flood messages to discover the topology and maintain the graph data structure, $G(V, E)$. Thereby, with the topology of the OpenFlow network known, the OpenFlow controller has an instance of the data structure of G and can manage the flows inside the network. One way to reach that topology information is fetching OpenFlow switches with control packets to discover the origin and destination of the links, switches and hosts (see Section 2.3.1).

4.2.2.3 HYDRA protection phase

The protection phase of HYDRA is responsible for distributing the primary and backup paths through the network. Algorithm 2 depicts the protection phase of HYDRA, where it is assumed that the network topology is known, the set of endpoints and a set of path identifiers is provided. Between lines 1 and 6, the primary path is computed using Dijkstra's algorithm [3] (line 2). Afterwards, a unique identification for primary and backup paths is reserved from the set of identifiers (lines 3 and 4). Those identifications are used in the construction of the paths using the OpenFlow with FF, and be a unique identification for each flow is a requirement of the protocol. After that, the identification of each flow is assigned to be installed for primary and backup paths (lines 5 and 6).

Algorithm 2: The algorithm of HYDRA for the protection phase.

```
Require: The network topology $G(V, E)$, set of endpoints $O$, set of identifiers $I$.
Ensure : For all endpoints, install primary and backup paths between source and
         destination.
1 for $(s, d)$ in $O$ do
2     $P_{sd}$ ← Find the shortest path from $s$ to $d$ on $G$ with Dijkstra's algorithm;
3     primary_id ← get_an_available_tag($I$,$d$) ;
4     backup_id ← get_an_available_tag($I$,$d$) ;
5     install_primary_path($P_{sd}$,primary_id) ;
6     install_backup_path($P_{sd}$,backup_id) ;
```

For installation of the primary path, Algorithm 3 is used. The algorithm requires the network topology, the identifier selected for the primary path, the switches that compose the endpoints and the path between those endpoints. Lines 1–4 initialize the variables, and then the installation of the primary path occurs between lines 5 and 10. The basic idea is to send for each OpenFlow switch in the path between the endpoints, the *OFPFlowMod*

[3] An algorithm for finding the shortest paths between nodes in a graph.

Algorithm 3: Install primary path.

```
Require: The network topology $G(V,E)$, an identifier $i$, the definition of flow $d$,
         the action to be applied $action$, the switch source $sw\_src$, the switch
         destination $sw\_dst$, and the $P_{sd}$ path from $sw\_src$ to $sw\_dst$
Ensure : Install a primary path between switch source and switch destination
         using the selected identifier
1  $in\_port \leftarrow 0$;
2  $out\_port \leftarrow 0$;
3  $counter \leftarrow 0$;
4  $path\_length \leftarrow ||P_{sd}||$;
5  for $counter < path\_length - 1$ do
6      $out\_port \leftarrow get\_port(G, P_{sd}[counter], P_{sd}[counter+1])$;
7      $match \leftarrow MT(i,d)$;
8      $opo \leftarrow create\_openflow(out\_port, match, action)$;
9      $send(opo, P_{sd}[counter])$;
10     $counter \leftarrow counter + 1$ ;
```

packet instructing the match (definition of flow d and the identification of flow i), the output port and the action.

Regarding the variable *action*, the "FORWARD"[4] value must be executed by the algorithm, once Algorithm 3 represents a flow creation process. If *action* is not "FORWARD", the first switch in the path, *sw_src*, receives an instruction to "DROP" the packets that match the *match* variable, and hence blocking the packets belong to that flow. For simplicity, this process of blocking packets was not included and is not shown in Algorithm 3.

Once the primary path is installed, the installation of the backup paths take place. The Algorithm 4 installs the backup paths for each switch in the primary path. The idea here is the use of *Proactive* install rules that avoid any unavailability in the primary path.

4.2.2.4 HYDRA restoration phase

After the HYDRA protection phase, the HYDRA resilience mechanism begins the restoration phase. Algorithm 5 depicts the restoration phase of the HYDRA approach. This phase is important to avoid the underutilization of flow entries in the OpenFlow network. Thereby, the algorithm uses the $ofps$ packet with the information of the link failure. Between lines 1–5, the involved switches are identified, the topology is updated with the failed link, and the affected flows are discovered.

Then, once the current network state is known, then new paths for the affected flows can be installed. Lines from 6–10 depict this process for each flow affected by the link failure.

[4] The proposed architecture adopted only two types of actions: forward and drop. See Subsection 4.1.3 for more information.

Algorithm 4: Install backup paths.

```
Require: The network topology $G(V, E)$, an identifier $i$, the definition of flow $d$,
         switch source $sw\_src$, switch destination $sw\_dst$, $P_{sd}$ path from
         $sw\_src$ to $sw\_dst$
Ensure : Install backup paths between switch source and switch destination using
         the selected tag.
1  for $src\_node$,$dst\_node$ in $P_{sd}$ do
2      $e \leftarrow$ get_edge($E$,$src\_node$,$dst\_node$);
3      $G_f \leftarrow G(V, E - e)$;
4      $P'_{sd} \leftarrow$ Find the shortest path from $src\_node$ to $dst\_node$ on $G_f$ with
        Dijkstra's algorithm;
5      $in\_port \leftarrow 0$;
6      $out\_port \leftarrow 0$;
7      $counter \leftarrow 0$;
8      $path\_length \leftarrow ||P'_{sd}||$;
9      for $counter < path\_length - 1$ do
10         $out\_port \leftarrow get\_port(G, P'_{sd}[counter], P'_{sd}[counter + 1])$;
11         $match \leftarrow MT(i, d)$;
12         $opo \leftarrow create\_openflow(out\_port, match, action =' FORWARD')$;
13         $send(opo, P'_{sd}[counter])$;
14         $counter \leftarrow counter + 1$;
```

Algorithm 5: The algorithm of HYDRA for the restoration phase.

```
Require: The network topology $G(V, E)$, set of $F$, set of identifiers $I$ and $ofps$
Ensure : The flows will follow the shortest path
1  $l \leftarrow get\_failed\_link(ofps)$;
2  $src\_node$,$dst\_node \leftarrow get\_switches\_affected(G, l)$;
3  $e \leftarrow$ get_edge($E$,$src\_node$,$dst\_node$);
4  $G_f \leftarrow G(V, E - e)$;
5  $affected\_flows \leftarrow get\_affected\_flows(G, l, F)$;
6  for $flow$ in $affected\_flows$ do
7      $src, dst \leftarrow get\_flow(G, flow)$;
8      $P'_{sd} \leftarrow$ Find the shortest path from $src$ to $dst$ on $G_f$ with Dijkstra's algorithm;
9      $primary\_id \leftarrow$ get_an_available_identifier($I$,$d$);
10     install_primary_path($P_{sd}$,$primary\_id$);
```

4.3 DYNAMIC OF THE PROPOSED ARCHITECTURE

This section describes the dynamic of the proposed architecture. First, it presents how a peering session is established between a domain with the proposed architecture and others traditional ASs, in Subsection 4.3.1.

Once the peering is established and the network traffic information is exchanged, the architecture can explore the multi-paths of the inter-domain environment, in Subsection 4.3.2. However, to avoid the creation of loops during the multi-path exploration, the dynamic of the proposed architecture also provides a loop avoidance mechanism, in Subsection 4.3.3.

4.3.1 Establishing a peering session

In a traditional TCP/IP network, when an AS has to set up a BGP peering with an other AS, a network operator requires to configure BGP routers with the peering information, such as ASN, IP addresses, whether multi-hop and others. Because COOL is a solution for OpenFlow networks with BGP compatibility, it is required to provide some peering information for the *BGP Speaker* component.

In Figure 27 depicts an example of how a peering session can be established. The basic peering information requires:

- peer router IP: the IP unicast value of the remote peer;
- ASN: AS number of the remote AS;
- switch_id: an unique identifier of a *border* OpenFlow switch;
- port_of_switch: the port that connects a *border* switch to the peer remote router;
- IP_controller: the IP unicast value of the OpenFlow controller;
- router_id: the identification of the OpenFlow controller.

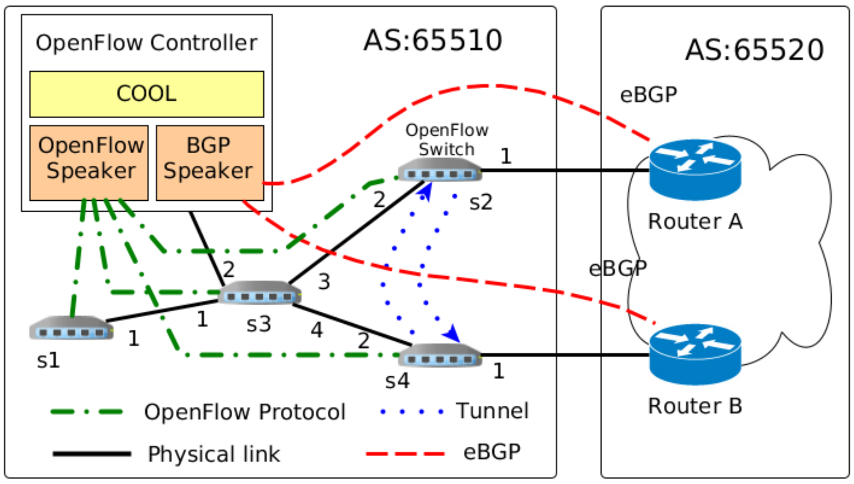


Figure 27 – An example of peering in COOL.

Thus, for neighbor peering, all configuration for peering between a neighbor and the domain that deploys the proposed architecture has to be provided by the network operator in the same way as the traditional BGP configuration. The OpenFlow controller has to be informed about the port where the *BGP Speaker* component will establish the peer session for a given neighbor. In addition to the port information, the ASN of the foreign AS and the IP address of the remote router have to be provided. Furthermore, if the IP address of an AS is a multihop, then the number of networks between the *BGP Speaker* and the router of the neighbor AS has to also be provided as by default, all IP packets from the *BGP Speaker* to a router neighbor is set with the value 1 in the Time To Live (TTL) IP field for an external BGP connection.

Backing to the Figure 27, once the peering is established, the OpenFlow controller knows what prefixes where learned from each peer. For example, supposing that a prefix from AS:65520 in the Figure 27 was learned through port 1 of switch *s4*. Thus, if that network prefix is the preferable, the main path to reach AS:65520 will outbound AS:65510 domain using port 1 of switch *s4*.

Then, if everything goes as planned, and the BGP peering is established, the exchange of NLRI between the peers can be made. The NLRI will compose the Adj_RIB_IN, and after applying the *Import Policies*, the LOCAL_RIB is populated. The LOCAL_RIB is generated by executing the BGP decision process. The RIB information is useful to the AS that deploys the proposed architecture as it can provide the other ASs multiple paths for traffic management that passes through the infrastructure. Thus, the proposed architecture explores the RIB information and provides more management control to customer ASs using inter-AS communication.

Furthermore, if the AS that deploys the proposed architecture allows a customer to manipulate the traffic in its network infrastructure, it will always be limited to the subset of prefixes learned and valid in the LOCAL_RIB. Besides, these prefixes have to be in the *Routing Registry* database and assigned to the same customer that requires traffic management. That is, the remote control of a customer will be limited to only a subset of the LOCAL_RIB, and a customer AS cannot change the forward network behavior of other prefixes that are not registered to it.

For the exchange control messages between a customer and a provider of the proposed architecture, it establishes a secure channel between the ASs (e.g., Virtual Private Network (VPN) or Hyper Text Transfer Protocol Secure (HTTPS)), which connects the AS that provides the services (for example, an ISP) to its customers' ASs. With a security channel, the network operator in the customer AS can exchange management information and monitor the traffic utilization destined to its network infrastructure.

Thus, the *AS app* of a customer AS will be responsible for manipulating how traffic is managed. The organization of the proposed architecture is aligned with one of the SDN principles, where the logic of the network occurs in applications executed over the controller. The *AS app* receives the BGP network state (extracted from the RIBs) for the valid prefixes and the network bandwidth usage for each path associated with each prefix.

When the network operator in the customer site receives the network state from its *AS app*, it will be able to configure the parameters of the execution of the *AS app*. This allows the network operator to manage how traffic of the inter-domain network will reach its network infrastructure. For example, the customer AS can send control information that describes how to load balance the traffic that passes through the AS that deploys the architecture, once it has access to the available bandwidth usage information for each prefix and the available and valid routes.

The default forwarding actions are to use the BGP behavior. With the information on how to treat the traffic, the AS that deploys the proposed architecture can apply the new control for a given network prefix and then override the BGP default behavior when required. Furthermore, both the customer and provided ASs can monitor the network traffic to verify if it has fulfilled the customer's requirements. Any changes in the BGP's view of the network that deploys the proposed architecture is rapidly notified to the customer via *COOL Interface* to take the appropriate action.

4.3.2 Exploring multi-paths

The architecture is capable of verifying if the multi-path of the domain can be explored. This is done by sending control packets through the paths learned and verified if those are valid and available. If this is the case, then the path can be used to forward traffic.

Suppose the example in the Figure 28. The stub AS A advertises the prefix 1.1.0.0/16 to its ISP X, then X propagates that prefix to its neighbors. The same process occurs with stub AS C that advertises prefix 2.2.0.0/16. Eventually all ASs in the topology have the routes to reach AS A and C.

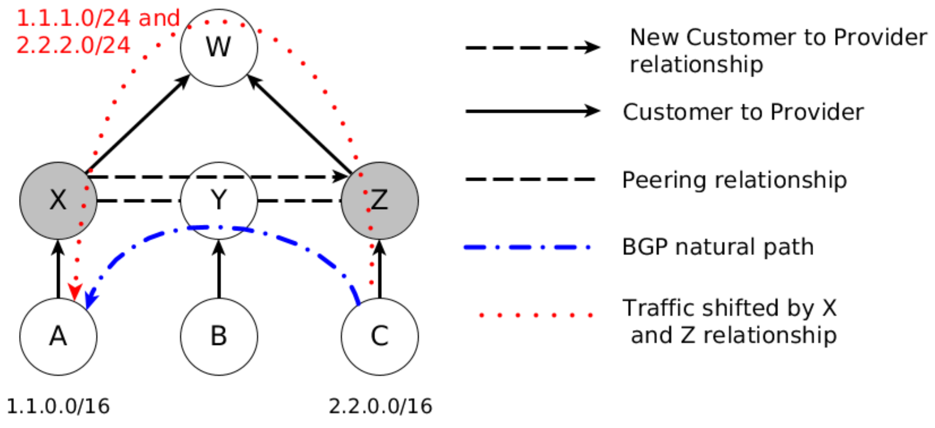


Figure 28 – Example of multipath scenario, in which circle represents an AS and the links between them the relationships.

Because the BGP follows the destination-based forwarding paradigm and considering the relationship between the domains in the Figure 28, a natural BGP path for traffic from AS C to AS A is created with the path be [C-Z-Y-X-A] (CARDONA et al., 2016). If ISP Z deploys the architecture proposed, AS X can become a new customer of ISP Z. In this case, the new Customer to Provider relationship can enable AS X to request shifting traffic from stubs AS C and A. Thereby, a subset of traffic from AS C to A can be forward to use path [C-Z-W-X-A] and the remain of traffic forward the BGP natural path, [C-Z-Y-X-A]. Besides, to avoid loops during the path diversity exploration, a loop avoidance mechanism is deployed in the architecture.

4.3.3 Loop avoidance

BGP only advertises its "best" path routes for its peers, hence, hiding the potential of exploring the path diversity. With a central entity peering with the domain's neighbors allows the AS (especially the ISPs) to know every possible exit and the routes reachable through its links. Those paths can produce a loop, and consequently decreasing the performance of the network. Loops can occur inside a network or during connections between networks.

For a network inside the AS, the loop avoidance system relies on the path computation based on information acquired from the SDN network. Multiple works already had been done about how to compute paths without loops in the context of SDN networks (NGUYEN et al., 2016; SILVA, 2018c). Indeed, the Section 4.2 depicts approaches for creation of flows inside the SDN networks that avoid loops, once it uses algorithms loop-free proved. Thus, the challenge for loop avoidance belongs to the inter-domain environment.

Because the proposed architecture is backward compatible with BGP, it relies on the BGP loop prevention mechanism to avoid loop during the application of the new types of routing policies in the inter-domain for single path approaches. Thus, *BGP Speaker* analyzes the BGP AS-Path attribute of a prefix learned, and if it finds the ASN of the domain in the AS-Path, it will ignore that prefix. That is the standard behavior of BGP to avoid loops for a single path, and possible transient loops can occur during the convergence of BGP using the AS-Path mechanism.

However, to explore multiple paths in simultaneous, it is fundamental to develop another loop avoidance mechanism. For example[5], supposing the topology representing some connected ASs as in Figure 29, if the AS 4 deploys the architecture and a customer AS 0 requires the explore the multi-paths for its domain, AS 4 could forward packets to AS 0 using paths: [5-2-0], [1-0], and [3-1-0]. Supposing the network state of the Figure 29 and that one of the neighbors of AS 4 (AS 5, AS 3 or AS 1), decides to prefer send traffic to AS 0 through AS 4. Let's say that it is AS 3. In this case, AS 4 will not use the path [3-1-0] because AS 3 will forward traffic back, and then, forming a loop between AS 4 and AS 3. To avoid this situation AS 4 must verify if the packets sent for each path will reach the destination.

The verification, if some paths are available or not, can be done by sending some control packets from the AS that deploys the proposed architecture to the customer AS.

[5] This example is based on a case scenario presented in the work of Qiu, Wang and Gao (QIU; WANG; GAO, 2007).

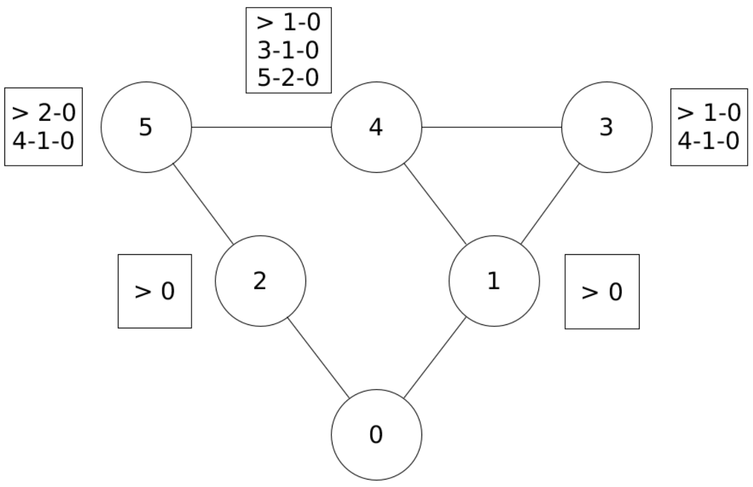


Figure 29 – Transient failure scenario in the inter-domain routing. The symbol ">" indicates the preference route to reach AS 0.

Thereby, AS 4 can send control packets to AS 0 for verifying the availability of path [3-1-0] and if those packets come back to the *BGP Speaker* or they are not acknowledged by the customer AS, in the case AS 0, then the path becomes unavailable to be used.

4.4 APPLICATIONS OF THE PROPOSED ARCHITECTURE

Some OpenFlow applications were conceived to test the applicability of the proposed architecture: three types of reactive load balancers and a scheme to protect primary routes.

4.4.1 Reactive Load Balancers

As explained by Silva (SILVA; DIAS; SADOK, 2017), there are different strategies for SDN load balancers. However, in this work, it was decided to develop three types of strategies for the load balancers to be used as the control application in the *AS app*:

1. Reactive Random Load Balancer (RLL);
2. Reactive Round-Robin Load Balancer (RRLL);
3. Reactive Round-Robin Load Balancer with a Threshold (RRLLT).

These three strategies use the reactive flow creation approach (SILVA, 2017) where the flows are created on-demand in the response of a packet that requires a new flow. The baseline for the comparative will be the default behavior of the BGP that uses the "best" next hop to the forward packets. The remainder of this Subsection is dedicated to depicting each approach.

4.4.1.1 Reactive Random Load Balancer with Random strategy (RLL)

A reactive load balancer creates flows on-demand. Consequently, when there is no OpenFlow rule installed in the data plane's devices that match against the incoming packet, the controller is notified. Thus, the work of a reactive load balancer with random selects a path from a list of valid and available paths by submitting that list to a Random algorithm. Then, it randomly selects one of those paths to compose a new rule for that flow. Algorithm 6 depicts the RLL.

Algorithm 6: Reactive Random Load Balancer

```
Require: The list of valid and available next hop to a prefix $L$, and a random
         function that selects an item from a list $random$.
Ensure : Select a next hop for a given destination.
1 next_hop ← $random(L)$;
2 return next_hop;
```

4.4.1.2 Reactive Round-Robin Load Balancer (RRLL)

Instead of a Random algorithm, a RRLL applies the Round-Robin algorithm to the list of available and valid paths. The Round-Robin scheduling algorithm is the simplest and easiest to be deployed once the first element in the list is selected and added in the tail of that list. This process is continuously repeated following the list in a circular order and selects each path in the list equally. Algorithm 7 presents the RRLL scheme.

Algorithm 7: Reactive Round-Robin Load Balancer

```
Require: The list of valid and available next hop to a prefix $L$.
Ensure : Select the next hop for a given destination.
1 next_hop ← $L.pop(0)$;
2 $L.append$(next_hop);
3 return next_hop;
```

4.4.1.3 Reactive Round-Robin Load Balancer with Threshold (RRLLT)

The RRLLT is an extension of the aforementioned strategy where a threshold of bandwidth usage is provided for each path in the list. Thus, if the bandwidth of a path overcomes this threshold, the algorithm must consider the associate path congested and not able to be used for flow creation until the usage becomes lower than the threshold value.

Algorithm 8 depicts the RRLLT algorithm. It requires a list of valid and available next hop to a given prefix (multi-path), the best next hop for the prefix (extracted from the BGP decision process), the outbound links contains the network state of bandwidth usage (main metric of the algorithm) and a threshold. Thereby, for each valid and available next hop in the list, the algorithm will verify if the next hop is congested or not by compare it against the threshold. If the next hop in evaluation is less than the threshold, then Round-Robin algorithm is applied and the next hop selected is used. However, all the outbound links for the a given prefix is congested, i.e., the bandwidth usage is greater than the threshold, then the best next hop is used. The idea of selecting the best next hop as the default next hop for a given prefix is just to define a criteria to the situation when all outbound links for that prefix is congested. Others criterion can be adopted to change the behavior of RRLLT (see in future works section in Section 6.2).

To be clear, all the already created flows assigned to the path that has overcome the threshold will continue to use that path, even when the path state is congested. This decision will avoid degradation in the transport level protocols that require keeping the state of established connections.

Algorithm 8: Reactive Round-Robin Load Balancer with Threshold

```
Require: The list of valid and available next hop to a given prefix $L$, the BGP
         best next hop *best* for the prefix, and the bandwidth usage list of
         outbound links in the domain $D$, a threshold for bandwidth utilization $t$.
Ensure : Select the next hop for a given destination.
1 length_of_L ← $get\_length(L)$;
2 for next_hop_id in length_of_L do
3     if D[next_hop_id] $\leq t$ then
4         next_hop ← $L.pop(next\_hop\_id)$;
5         $L.append$(next_hop);
6         return next_hop;
7 return best;
```

4.4.2 Protecting primary routes

For brevity, the notation is summarized in Table 10 with an individual description of each item. This notation is used to compose the protection scheme for primary routes. Furthermore, this work models an OpenFlow network as a directed graph (or digraph) $G(V, E)$, where V is a set of nodes (e.g., OpenFlow switches) and E a set of edges (e.g., network links).

The topology of the OpenFlow network is assumed to be known, and the OpenFlow controller has an instance of the data structure of G. One way to reach that topology information is fetching switches with OpenFlow control messages to discover the origin and destination of the switches, links, and hosts.

Once with the topology state, to protect the network traffic against external link failures, it is required that the the solution generate additional flows to protect the primary routes. The idea here is to use proactive rules into the OpenFlow network infrastructure and let the OpenFlow switches handle the connectivity restoration when an external link

Table 10 – Notation adopted

Notation	Description
$G(V, E)$	The network topology, where V denotes the set of nodes (switches) and E the set of edges (links between switches)
$s(\bullet, \bullet)$	The function which returns the border switch of a given prefix.
R	The set of routes in the Local RIB of *BGP Speaker*.
sw_s	The border switch where the primary path ends.
sw_d	The border switch where the alternative path ends.
$S_{\bullet,\bullet}$	A path from source $\bullet$ to destination $\bullet$.
$t_\bullet$	A tag identification for the tunnel.
$ff_\bullet$	A tag identification for the FF.

Algorithm 9: Protecting algorithm

```
Require: The network topology $G(V, E)$, the local RIB $R$, and the set of
         alternative routes $A$.
Ensure : For all primary route, calculating a tunnel path from the OpenFlow
         switch port where the route outbound to an alternative outbound link of
         another border switch, and installing the new tunnel.
1 for $r$ in $R$ do
2     $sw_s \leftarrow s(r, P)$ ;
3     if $r$ in $A$ then
4         $sw_d \leftarrow s(r, A)$ ;
5         $S_{p,a} \leftarrow$ Find the shortest path from $sw_s$ to $sw_d$ on $G$ with the Dijkstra's
            algorithm;
6         $t_{id} \leftarrow$ get the tunnel identification;
7         $ff_{id} \leftarrow$ Create an unique identify for the FF;
8         Construct a tunnel path using $S_{p,a}$, $t_{id}$ and $ff_{id}$;
```

is not available. Thus, all traffic will be redirected to an alternative outbound link, and all prefixes that are reachable to the domain through more than one external link will be recovered.

The algorithm for protecting routes is depicted in Algorithm 9. It requires the set of primary routes R that is extracted from the *Local RIB* component (see Figure 24), the network topology G extracted from the *Topology Management* module and the set of alternative routes A extracted from the *Valid and Available Paths* component inside the *Topology Management* module. With those information, for each primary route r of the local RIB R, and an alternative path is computed to connect the switch border of the primary path to the switch border of the secondary path.

4.5 CONCLUDING REMARKS

This Chapter tackles the network control in inter-domain routing and proposed a new architecture to manage traffic to multi-homed ASs where an AS using SDN technologies that can provide new services and allowing its customers to apply new network management tasks into its infrastructure. It also explores the challenges of managing flows inside the SDN domain, where multiple algorithms are depicted to warranty the resilience for the flows in the SDN network.

The dynamic of the proposed architecture is detailed, including how to establish a peering session between domains, how the architecture explores the multi-paths available in the inter-domain environment, and how it avoids loops during its execution. Finally, the Chapter finished describing three applications to be used in the evaluation of the proposed architecture. The next chapter explores the evaluation of the architecture depicted in this chapter.

5 EVALUATION

This evaluation Chapter is divided into two parts. The first part is dedicated to evaluating the COOL controller that includes: resilience mechanisms for OpenFlow networks; and, avoiding inconsistency in flow creation. The second part is dedicated to evaluating the proposed architecture that includes: manage inter-domain traffic using a proof of concept scenario; the protection against external link failure using the FF protection mechanism of OpenFlow; and, an investigation about traffic exchange messages. And, at the end of each part of this Chapter, and discussion about the evaluation made is performed.

5.1 EVALUATION OF THE COOL CONTROLLER

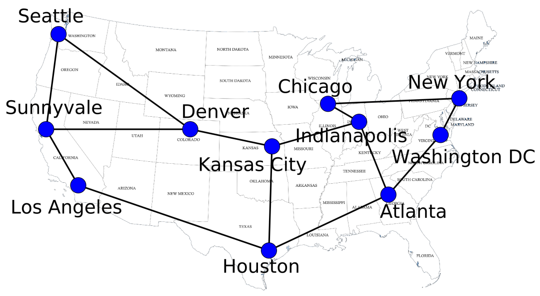
The COOL controller is based on the Ryu controller (RYU, 2016) [1] and publicly available on the site: <https://github.com/walberjose/COOL>. COOL uses the OpenFlow 1.3 version for its *OpenFlow Speaker* and version 4 of the BGP protocol in the *BGP Speaker*. Furthermore, multiple modules were developed to embrace the requirements of the proposed architecture depicted in Chapter 4. Therefore, this section evaluates the COOL controller regarding resilience mechanisms and the mechanism to avoid inconsistency in stateful applications.

5.1.1 Resilience mechanisms for OpenFlow networks

The COOL controller implements multiple resilience mechanisms for OpenFlow networks. For performance evaluation, this subsection is dedicated to analyze the HYDRA (presented in details in Subsection 4.2.2 of Section 4.2) against the PP, LR, PR and LFR approaches. The SFC can be used as a reference in the experiments once it is a simple way to create flows in an OpenFlow network. Just remembering, PP is a protection resilience mechanism (that creates backup rules in advance), and LR, PR and LFR are restoration approaches (see Section 2.4 for more detail about those resilience mechanisms). The description of those algorithms, as well as the literature reviewed are in Section 2.4.2.

Two computers were used to emulate the OpenFlow network scenarios. The first computer was a dedicated machine with Ubuntu Version 17.04, 8 CPU cores with a clock of 2.20 GHz and 8 GB of RAM. This computer was used as the OpenFlow controller with Ryu (RYU, 2016). The second computer has Ubuntu Version 14.04.4 with 3 GB of RAM and a dual-core CPU with a 2.4-GHz clock. This second machine executed Mininet (MININET, 2016) for the network emulation environment. A physical Ethernet cable of 100 Mbps connected the computers in crossover mode.

[1] Ryu is a component-based SDN framework written in Python language.



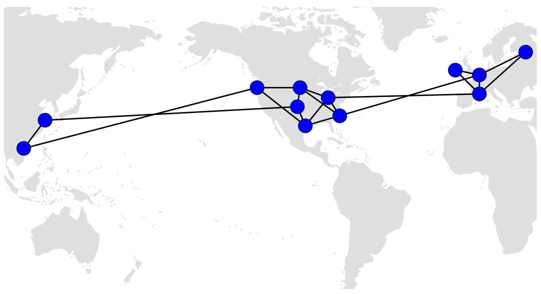


Figure 30 – Topologies adopted in this work. (**a**) The representation of the Abilene topology. (**b**) The representation of a WAN Google topology.

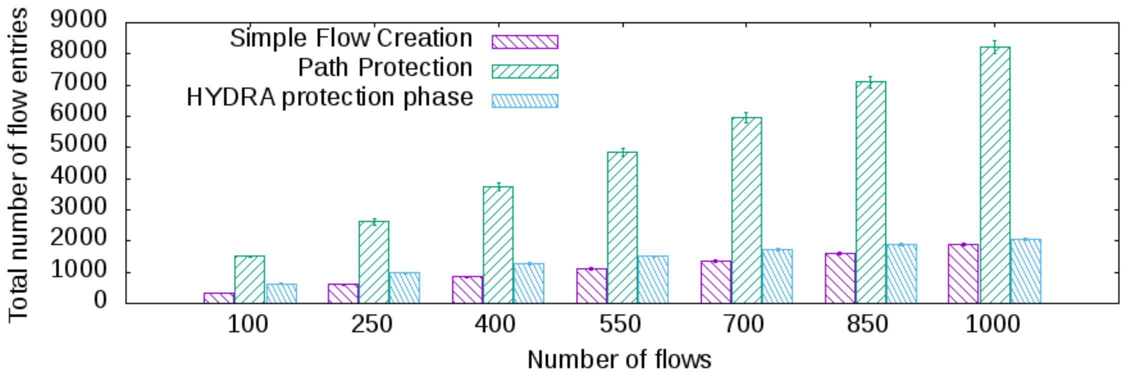
The dataset topology from Internet Topology Zoo (TOPOLOGY-ZOO, 2017) was used. Topology Zoo is an ongoing project that collects network topologies' information from around the world, and it is very useful for researchers to access information from real-world topologies. The 2005' Abilene topology was adopted to apply the experiments of HYDRA. Once, with the data network information, the network topology was reproduced inside the emulation environment of Mininet (MININET, 2016). The topology used has 11 nodes and 14 links. Figure 30a presents the Abilene topology representation. The bandwidth links were 1000 Mbps with zero delay value and none packet loss during the execution of the experiments using the Mininet.

Besides the Abilene topology, another topology adopted was from the Wide Area Network (WAN) Google topology. The Google topology was extracted from the work of (JAIN et al., 2013). It was composed of 12 nodes and 18 links, and its representation is in Figure 30b, and adopting the same configuration of Mininet used in the Abilene topology's links. Therefore, all topologies adopted in this work are in Figure 30.

5.1.1.1 Number of flows entries

One way to evaluate HYDRA, executing in COOL, is using the total number of possible flow entries required for connecting two endpoints in the OpenFlow network. The flow entries can be understood as the rules used in the flow table of an OpenFlow switch. Thereby, the flow entries consumed for the approaches measured were: SFC (where a flow connects an endpoint in the topology); PP; and HYDRA protection phase (for details about the last two approaches, see Section 2.4.2).

The endpoints (composed of a source and a destination) were randomly selected in one of the topologies and the number of flow entries in each experiment (counting all network rules installed in the OpenFlow switches) executed was measured more than 300 times. After that, the average of the number of flow was computed. Besides, for the Abilene and Google topologies, each experiment was configured as 10 flows per endpoints. Thus, increasing the number of endpoints used in the topology will also increase the number of rules used in the OpenFlow switches.



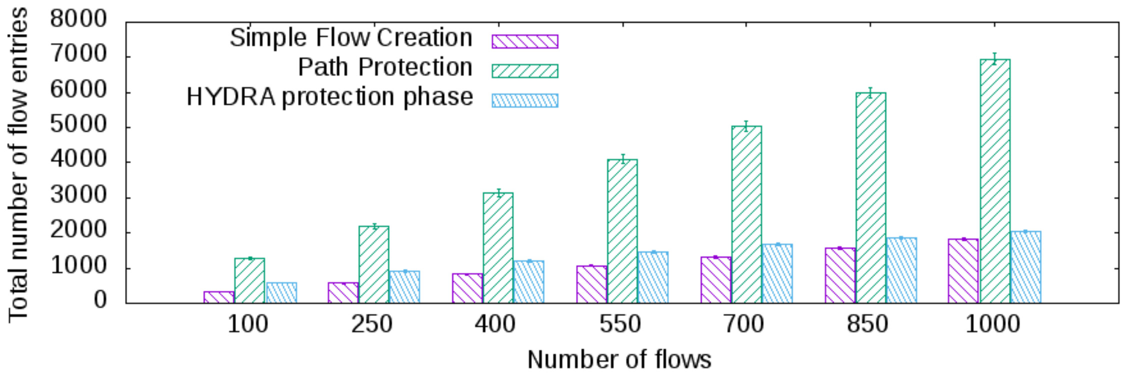


Figure 31 – Number of flow entries for different topologies. (**a**) To the Abilene topology, the number of flow entries for each approach adopting 10 flows per endpoints. (**b**) The number of flow entries for each approach with 10 flows per endpoints using the Google topology.

In total, the number of endpoints possible in the Abilene topology is 110 and for the Google topology 132. For convenience, the selected number of endpoints were [10, 25, 40, 55, 70, 85, 100], resulting in the number of flows generated in the topology of [100, 250, 400, 550, 700, 850, 1000], respectively. After collecting the measurements, the average and standard deviation were calculated, and confidence intervals at the confidence level of 95% was used. The results for the Abilene and Google topologies are in Figure 31.

The baseline for analysis is the SFC, and this algorithm uses one flow entry in each OpenFlow switch for every flow created for an endpoint. Compared it against PP, PP consumes much more flow entries on average to connect the endpoints in both topologies; see Figure 31a,b, respectively. This result is expected as for each flow, a protection path must be created, and that protection path avoids the unavailability of the flow in the presence of a link failure.

However, with the HYDRA protection phase, the average number of flow entries is higher than SFC for each quantity of number of flows evaluated. This difference is caused by the additional use of flow entries to represent the backup tunnel paths required to protect the main path of each flow. Compared to PP, the HYDRA protection phase is a more scalable solution based on the results achieved and the topologies adopted because it will consume the least number of flows on average.

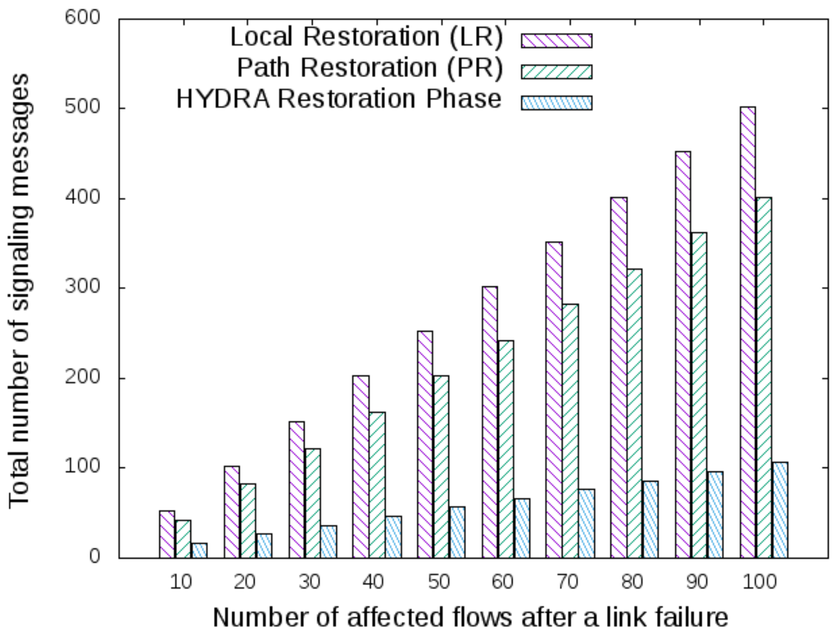


Figure 32 – Results for the signaling overhead after a link failure in the Abilene topology.

5.1.1.2 Signaling overhead

A criterion for measurement scalability is using the number of messages as an evaluation metric (MENDIOLA et al., 2017). Protection approaches install primary and backup flow entries in advance to prevent connectivity interruptions. After a link failure, no signaling messages between OpenFlow switches and the controller are sent to restore connectivity. This is true if and only if the rules use FF (or other hardware-based resilience solution). Otherwise, the number of signaling messages between switches and the controller is the number of flow entries affected by the failure, because it is necessary to change the forward port number for every flow entry affected by the failed link.

Hence, it is mandatory to investigate the signaling overhead of resilience mechanisms. Those are analyzed using scenarios that describe a link's failure occurring in one of the topologies adopted. Thereby, the Abilene topology depicted in Figure 30a will be used, and the following resilience mechanisms will be compared: LR, PR and HYDRA Restoration Phase.

For example, suppose that in the topology adopted (Abilene topology; see Figure 30a), the primary path is made through the cities of Seattle, Sunnyvale, Los Angeles and Houston (any other path in the topology could be used). Suddenly, the link connecting Los Angeles and Houston suffers a failure (this link was choose because belongs to the primary path). The number of affected flows and the total number of signaling messages to recover from that failure are depicted in Figure 32. Regarding the LR and PR approaches and based on the achieved data, PR generates fewer signaling messages to the SDN controller than LR. This outcome is the opposite of the results presented in the work of Zhang et al. (ZHANG et al., 2016), where LR generated fewer update flow entries. The justification for this result is that the LR may not be the shortest path between two given endpoints.

Thereby, when a link in the path between two endpoints suffers a failure, if the shortest path from the OpenFlow switches' source and the destination of the failed link does not belong to the shortest path between the endpoints, then LR will require more signaling messages than PR. The reason for this signaling overhead is that there are more OpenFlow switches to restore the network connectivity after a link failure. Besides, LR and PR produce much more signaling messages with a relative lesser number of affected flows after a link failure, as a typical link requires hundreds of thousands of flows per link (e.g., in the IXP environment (GUPTA et al., 2016)). Regarding HYDRA, it notoriously overcomes LR and PR for signaling overhead criteria, since HYDRA Protection Phase already installed restoration rules and, after a link failure, signaling rules are required to restore an optimal state of the paths (shortest paths).

5.1.1.3 Failure recovery time

For the restoration approach, PR and LR were used. Both restoration approaches require signaling to the SDN controller to recover from a link failure. However, this is not the case for the protection mechanisms, which is the case of PP and the HYDRA protection phase, as all recovery rules are already installed inside the OpenFlow switches, and those rules protect the network from a link failure.

To execute the experiments, a subset of the Abilene topology (depicted in Figure 30a) was used. Using the iperf (IPERF, 2016) tool, a different number of flows from the node Sunnyvale (of Figure 30a) to the node Houston was generated, making the primary path composed of the nodes Sunnyvale, Los Angeles and Houston. Because this work uses the number of hops as the main metric to define what is a better path, the backup path is composed by the nodes Sunnyvale, Denver, Kansas City and Houston. Then, the link connecting Los Angeles and Houston suffers a failure, and for each resilience mechanism, the results of the failure recovery time are measured and plotted in Figure 33.

Figure 33 shows the failure recovery time, where the x-axis represents the number of disrupted traffic flows and the y-axis the failure recovery time in milliseconds for each resilience mechanism. As expected, the restoration approaches take more time to recover the flows, because a communication between the OpenFlow switch and the controller is required. However, PR reduces the restoration time against LR because it uses less nodes to signaling. PR and LR recover each flow individually, and hence, that behavior affects the recovery time.

Analyzing the protection approaches, because PP and the HYDRA Protection Phase already had installed the backup path using the FF. When a link failure was detected by the OpenFlow switch, then it automatically switched the primary path to the backup path. As a consequence, the recovery time becomes minimal. However, the number of

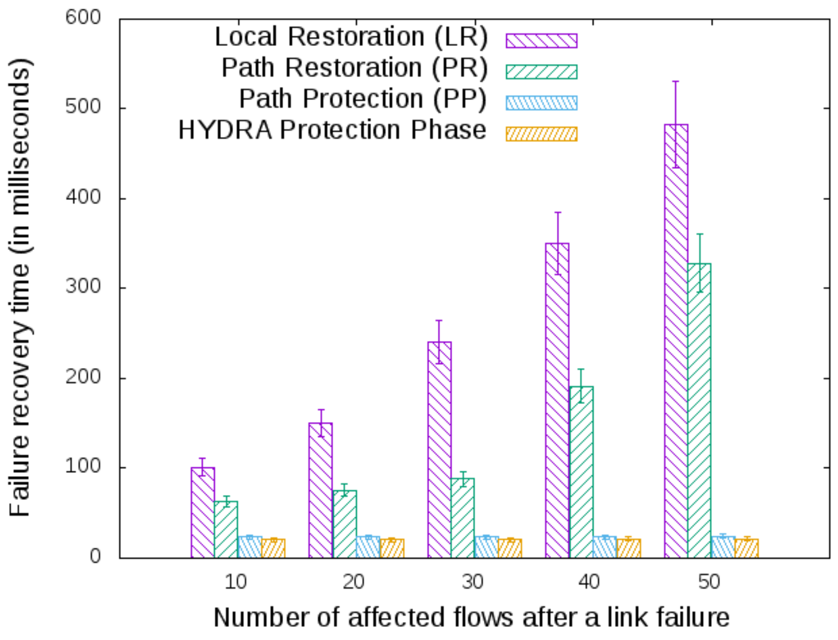


Figure 33 – Results for failure recovery time after a link failure in the Abilene topology.

flow entries required by the PP approaches is higher than the HYDRA protection phase (see Figure 31a), and that factor increases the lookup time of the OpenFlow switch table, affecting the failure recovery time of the flows for the PP.

5.1.2 Avoiding inconsistency in flow creation for stateful applications

This subsection evaluates the proposed scheme for avoiding inconsistency in flow creation for OpenFlow networks depicted in Section 4.2.1. Thereby, a Load Balancer (LB) application was developed to verify the applicability of the proposed scheme. Additionally, the state consistency of the LB application using the proposed scheme is also investigated. That is it, a stateful LB was developed to test the ideas for avoiding inconsistency in flow creation. The concept of a load balancing is to distribute load among various elements and improve the performance by optimizing the metric of interests (KHIYAITA et al., 2012). A LB is frequently used to distribute the load between different resources, or components of a system. Thus, to verify how the proposed scheme is applicable in practice, a LB program was developed to serve as a tester.

The RRLL[2] was chosen for the LB. Thus, the LB distributes the load of create flow in an OpenFlow network systematically, following the Round-Robin Algorithm (RRA), in which the identification number of all available OpenFlow switch ports are stored in a circular queue. The selection process uses the values from the head to the tail of the queue to select the forward port of a new flow and then starting again at the top of the queue. Thus, packets belonging to a new flow must be forwarded consistently to the port

[2] Reactive Round-Robin Load Balancer (RRLL)

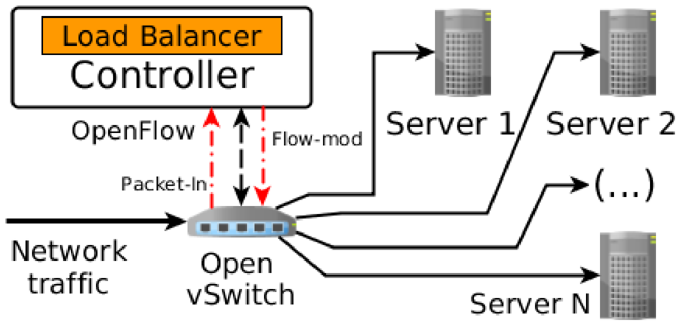


Figure 34 – Scenario description used in the evaluation.

that was previously assigned by the selection process, and this will only occurs if the LB receives the number of *Packet-In* events equivalent to every new flow produced.

Because RRA is a stateful algorithm, the LB with RRA is a stateful application as well. Here, LB works with the flows defined as a 5-tuple of IP source, IP destination, IP protocol, UDP source port and UDP destination port. All flows are distributed equally to the number of interfaces that connect to servers.

5.1.2.1 Scenario description

Figure 34 depicts the scenario. The Mininet(MININET, 2016) was used to create the emulation environment for the experiments. The virtual scenario executed in a dedicated machine with Ubuntu 16.04 LTS, 8 GB of RAM and 8 CPU cores with a clock of 2.20 GHz. Furthermore, the network traffic was generated using the version 2 of *iperf* (IPERF, 2016) tool, in client mode. The *iperf* produced traffic with different bandwidth rates when setting in the client's mode and using the UDP protocol. The number of datagrams generated were 71332 for every measurement, with 100 Mbytes for transfer (using the flag "-n 100M" of *iperf*) and the UDP buffer size of 208 Kbytes (default size).

The different bandwidth rates of *iperf* were used as well. Thus, the workloads used a fixed quantity of UDP packets sent for the *Load Balancer* application, however using different bandwidth rates. The rates used were 1Mbps, 10Mbps, 100Mbps, 200Mbps, 500Mbps, and 1000Mbps, and those values were chosen by convenience. For the experiments, the Open vSwitch (OPENVSWITCH, 2016) (with OpenFlow version 1.3 protocol enabled) behaved as the LB with RR, which handled the incoming network traffic and distributed to the servers directly connected (Figure 34). Besides, when a packet did not match any of the previous rules installed in the switch, the default action was to encapsulate the packet with a *Packet-In* packet and then send it to the controller.

After that, the controller treats the *Packet-In* packets with and without the proposed scheme as described in Section 4.2.1. It measured the number of *Packet-In* events that the LB application received during the flow creation process. After the LB decides how

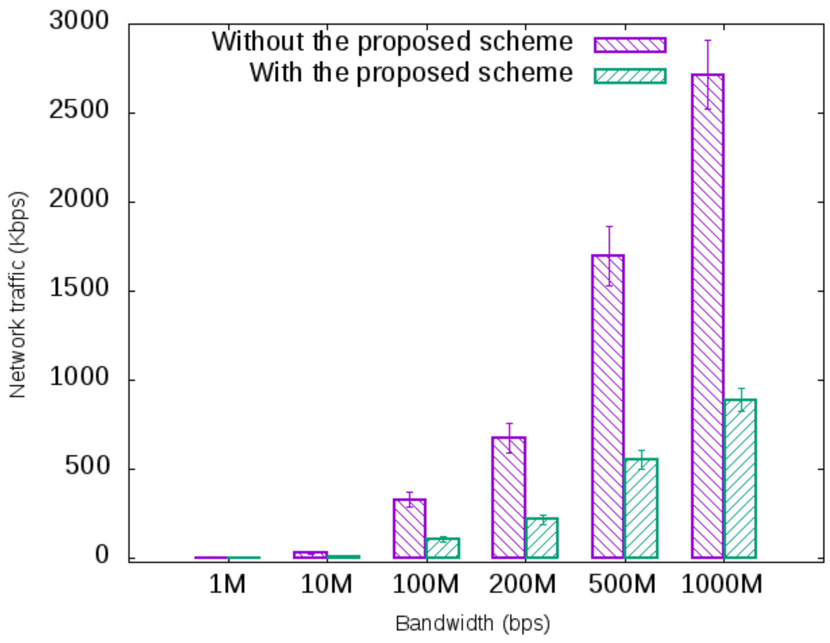


Figure 35 – Bandwidth usage between an OpenFlow switch and the controller.

the packet in the *Packet-In* will be handled, the LB instructs the controller to install rules in the OpenFlow switch through a *Flow-mod* packet. When the rules are installed, the packets are forwarded to one of the servers (Server 1, Server 2, ..., Server N) that executed the *iperf* (IPERF, 2016) tool in the server mode. Afterwards, it was measured the number of *Packet-In* events that LB received without and with the proposed scheme.

5.1.2.2 Results

The first result verifies the bandwidth generated by multiple *Packet-In* packets between an OpenFlow switch and the controller executing the LB application. When applying restrict values of “max_len” to the flow creation of the LB, following the *Definition of Flow*, there is an expectation of a reduction in the bandwidth usage.

Figure 35 depicts the bandwidth usage with and without the proposed scheme. All the values were measured using the Wireshark(WIRESHARK, 2017) with more than 100 repetitions, and a confidence interval of 95% was applied. For low bandwidth, the use of bandwidth with and without the proposed scheme is practically imperceptive. Nonetheless, with high rates, such as 1000 Mbps, the use of bandwidth is notorious.

Another result was the number of *Packet-In* events sent to the LB application for every workload adopted. The results are displayed in Table 11. This table shows the total number of *Packet-In* events received by the LB application using the proposed scheme and without. The experiments were executed more than a 100 times. The values in Table 11 represent the minimum, average and maximum number of *Packet-In* events measurements with the confidence level of 95%.

Table 11 – The average of *Packet-In* events reaching the stateful application with the confidence intervals for a confidence level of 95% (min/average/max).

Bandwidth	Without the proposed scheme	The proposed scheme
1 Mbps	1.0/1.0/1.0	1.0/1.0/1.0
10 Mbps	5.9/6.0/6.1	1.0/1.0/1.0
100 Mbps	526.8/559.5/592.2	1.0/1.0/1.0
200 Mbps	1218.3/1295.0/1371.7	1.0/1.0/1.0
500 Mbps	3312.6/3467.6/3622.6	1.0/1.0/1.0
1000 Mbps	3950.6/3983.3/4016.0	1.0/1.0/1.0

When the bandwidth rate is relatively low (< 1 Mbps), the controller with and without the proposed scheme passes the same number of *Packet-In* events for the application during a new flow creation. However, when a higher rate was used ($>$ 1Mbps in the experiments), the number of *Packet-In* events distributed to the application started to produce inconsistent in stateful SDN applications that relies on the count of *Packet-In* events. This behavior does not occur when the proposed scheme is used.

The multiple *Packet-In* events received for an stateful application executing in a controller without the proposed scheme can affect the logic behind the application, taking it to an inconsistent state. For example, the consequence for the LB with RR algorithm receiving inconsistent information from the controller was to unbalance the distribution of flow creation through to the servers, which is not the desired behavior.

However, when the experiments are executed with the proposed scheme, the number of *Packet-In* events sent to the LB becomes stable, and all *Packet-In* events generate a unique flow from LB. As a result, for all workloads, there is only one *Packet-In* event reaching the application when a new flow request arrives in the OpenFlow switch. Hence, for each request of a new flow, the LB can distribute the creation of flows without the need to treat multiple *Packet-In* events, therefore, keeping the consistency of the RR algorithm inside the LB.

One last observation of the Table 11 is the proximity of the measurement values of *Packet-In* events with and without the proposed scheme when the bandwidth usage rates are 500Mbps and 1000Mbps. This proximity occurs because of the overload in the controller, which started to drop packets due to CPU overhead.

5.1.3 Discussion about management of OpenFlow rules

SDN is a recent concept to how network engineers design networks and one of the challenges in OpenFlow networks is how to manage the rules to be used in the OpenFlow devices. Thereby considering this challenge and the proposed architecture, this section discusses the number of network rules affected by the proposed architecture in the inter-domain environment, the installation of rules in OpenFlow networks and the mitigation of inconsistency in SDN stateful applications.

5.1.3.1 Restrictions of the number of network rules

It is evident that exploring multiple paths of customers prefixes can increase the demand for OpenFlow rules. For example, in the topology adopted (see Figure 37), the execution of a load balancer application increased $n+1$ times the number of OpenFlow rules required to forward a given customer prefix, where n is the number of source network prefixes needed to be managed by the transit AS. Thus, instead of just one forward rule to the prefix 172.16.0.0/24 to the AS 400 (the default "best" next hop of BGP rule), 51 OpenFlow rules were required to manage the inbound traffic for the customer AS from the 50 prefixes of the *50 Networks*.

Regarding the number of network rules, the OpenFlow protocol required multiple match fields to compose a single rule, and the OpenFlow switches often deployed a flow rule using the TCAM. TCAM memories are a high-speed memory, but with the limitation of very high monetary costs. Moreover, TCAM requires a lot of space in its chip hardware and has a notoriously high energy consumption. Thereby, due to the fine-grain of the OpenFlow rules, these rules can be cumbersome to an inter-domain environment where the number of prefixes (or flows) is tremendously high (WANG et al., 2016a). Thus, it is worth noting that any OpenFlow based solution requires appropriate flow management. The rule placement problem inside an OpenFlow network is further detailed in the excellent and extensive survey of Nguyen et al. (NGUYEN et al., 2016).

Furthermore, OpenFlow specification, and hence the OpenFlow switches, aims to make the data plane elements flexible and efficient as Application Specific Circuits Integration (ASCI) used in the BGP routers (traditional network approach). However, to increase the flexibility of the OpenFlow protocol, the number of TCP/IP protocols supported for each new version of the OpenFlow protocol stumbles into the TCAM limitations. Therefore, a trade-off between flexibility (e.g., number of match fields) and performance (for example, number of rules) must be reached. Ideas for allowing OpenFlow data plane elements to be more flexible and programmable are at the very beginning such as P4 (BOSSHART et al., 2014) or OpenState (BIANCHI et al., 2014) and can be an answer for the flexibility and performance of OpenFlow switches.

5.1.3.2 Rules installation for OpenFlow networks

With regards to reducing the number of rules installed into the SDN network, the works of *Palette* (KANIZO; HAY; KESLASSY, 2013) and *One Big Switch* (KANG et al., 2013) are proposals that tried to addressed the OpenFlow switches' memory limitation. They considered that the rules to be installed are non-reducible, so they can not enforce rule aggregations. Thus, the solutions distributed the routing rules in the network in such a way that the routing semantics are maintained, and the network policies are not violated.

The work of (NGUYEN et al., 2014) also seeks to optimize the placement of routing rules within an SDN network. It accomplishes this by minimizing the resources required for the treatment of network flows. With an algebraic model and using the Integer Linear Programming, an optimization technique, to express constraints in the end-to-end routing policy on the network, the work indicated how to allocate a greater amount of traffic over memory capacity constraints using the model proposed. The work also performs comparisons with the solutions *Palette* and *One Big Switch* and found similar values of optimization. However, it overcomes the previous works in a scenario of extreme memory shortage, when the SDN controller must be triggered to maintain the minimum network operating state, even if the network performance degradation occurs.

Furthermore, to reduce signaling overhead between OpenFlow switches and OpenFlow controllers, it is necessary a prediction, or estimation of the network traffic, to allow rules installation in advance, and before the traffic ingress into the network. The accurate achievement prediction requires data collections and induces signaling messages, which makes this a difficult task.

For increasing the utilization of TCAM space and avoid TCAM misses, the authors in (VISHNOI et al., 2014b), presented a system that combines an adaptive heuristic with proactive eviction by choosing the timeout values of OpenFlow rules. They found that over particular types of network, with the understanding of the network traffic, is possible to outperform static timeout policy (fix value of OpenFlow timeout rules). However, to increase the utilization of the data plane elements' memory, it requires a heavy signaling overhead to the controller, because of frequently fetching information about the switches and flows states.

Thus, a trade-off among rule installation and the signaling overhead between OpenFlow switches and OpenFlow controller must be made. The work in (NGUYEN et al., 2016) classified the flow creation for OpenFlow networks into two categories: reactive, the rules are created on demand to react upon flow events; proactive, rules are populated in advance, that way the flow is created before the packet arrives at an OpenFlow switch port. One other option is to use the bird's eye of SDN to actively create network rules when the controller already discovers the path that the packet will take (SILVA, 2017).

5.1.3.3 Discussion about avoiding inconsistency

Regarding avoiding inconsistency in stateful applications, this work tackled the guarantee in the consistency of *Packet-In* events to the stateful application. Besides, manipulating OpenFlow rules' parameters, such as the "max_len" value in the action rules generates by the *Packet-In*, can bring benefits for the network, e.g., the reduction of bandwidth usage between switches and controller. This reduction is important to avoid saturation of the controller with useless information in the *Packet-In* packets when creating new flows. The results indicate that the bandwidth usage represents less than 1% (verified in all experiments) of the total available bandwidth between a switch and a controller. Despite that small percentage, this reduction can play a significant role when applied in a different scenario in which bandwidth is a critical resource, for example in Software Defined Wireless Networks (MIZUYAMA; TAENAKA; TSUKAMOTO, 2017).

However, turning OpenFlow rules' parameters is not appropriate for usage with all types of stateful applications. For example, a Deep Packet Inspection (DPI) application needs all data that a packet carries, and not a subset of those values (as setting "max_len" for reducing the overhead communication between data and control plane). In this scenario, for DPI, the recommendation is to use the value of OFP_NO_BUFFER (ONF, 2017), and then the whole packet will be sent to the controller for inspection.

Furthermore, the modification of *Event Engine* is sufficient to prevent a stateful application from changing its state several times in response to multiple incoming packets that belong to a unique flow request. The option to make the change in the *Event Engine* of the controller, and not to in all stateful applications that require a consistent delivery of *Packet-In* events, is to prevent every stateful application from developing its own control mechanism.

As concerns the experiments, the results are relevant for other researchers that use virtualization environment. For example, if the *Load Balancer* is submitted to a small amount of load (less than 1Mbps), it is possible that the wrong outcome of the distributed traffic may not be perceived (KOERNER, 2012; SILVA; SADOK, 2017). However, when the rate of incoming packets increases, the state storage in the stateful application may suffer changes that do not represent the desired behavior of the application. Hence, this scenario may become problematic for researchers to interpreting the data from a stateful application, because that application will be in the inconsistent state.

Regarding the related works, the values in Table 11 are consistent with the number of *Packet-In* events measured in the work of Mizuyama et al. (MIZUYAMA; TAENAKA; TSUKAMOTO, 2017), which is an indication that the measurements of this work were conducted appropriately. Moreover, here, the proposal was to modify only the controller and perform some control in the switches. The idea is to avoid having different approaches for new additional features in OpenFlow networks, such as the use of various flavors of OpenFlow specification or modified switches. Those different approaches can lead to incompatibility problems and make integration with preexisting software and hardware, or new ones, a difficult task.

For future works, other approaches can be tested to avoid inconsistency in stateful applications. One of them is the use of some programmable data planes frameworks such as OpenState (BIANCHI et al., 2014) or P4 (BOSSHART et al., 2014). Those frameworks attempt to develop flexible hardware that is as efficient as Application Specific Integrated Circuits

(ASICs). Thus, the hardware of the switch will be responsible for sending consistent control messages to the controller.

Finally, one caveat of the proposed approach to avoid inconsistency in the stateful applications is that there are some individual cases in which the number of *Packet-In* packets do not need to be controlled, and they are useful to applications. For example, to produce an estimation of how much bandwidth a flow will consume, the work of (MIZUYAMA; TAENAKA; TSUKAMOTO, 2017) counts the number of multiple *Packet-In* messages that a controller received to create new flows. Then, with the estimative of bandwidth for each flow, the controller could allocate flows in order to avoid network traffic congestion in the switches links.

5.2 EVALUATION OF THE PROPOSED ARCHITECTURE FOR MANAGING INTER-DOMAIN TRAFFIC

Once the capabilities of the COOL controller were investigated, this section evaluates the proposed architecture regarding managing inter-domain traffic, the protection of links against external link failures and traffic exchange control messages.

To perform the evaluation, a common topology is extracted from the Internet topology information, and afterward, a prototype network based on that topology is depicted. Then, the definition of workloads to be used in the prototype is established, following by the execution of the prototype that explores the management of inbound traffic using different strategies. Adopting the same prototype, another scenario investigates the situation when an external link fails. This section finishes with a study of the signaling overhead of the proposed architecture.

5.2.1 Manage Inter-domain Traffic

The goal of this evaluation is the execution of the proposed architecture to control inter-domain traffic. More specifically, the inbound traffic of a stub AS. This can be done by exploring the available outbound links of a given ISP to a prefix destination.

Besides, the methodology for evaluating the application of multiple strategies for load balancing traffic is similar to the work of Silva, Dias and Sadok (SILVA; DIAS; SADOK, 2017). First, the evaluation scenario is depicted, a prototype environment is built, workloads are defined and executed, and the results are presented.

5.2.1.1 Topology adopted

To adopt a topology for composing the evaluation scenario, the CAIDA's AS relationship database (CAIDA, 2016) was used for the characterization of multi-homed stub ASs. It was expected that those types of ASs get more advantages of the proposed architecture once it can allow them to control and explore the path diversity of the Internet.

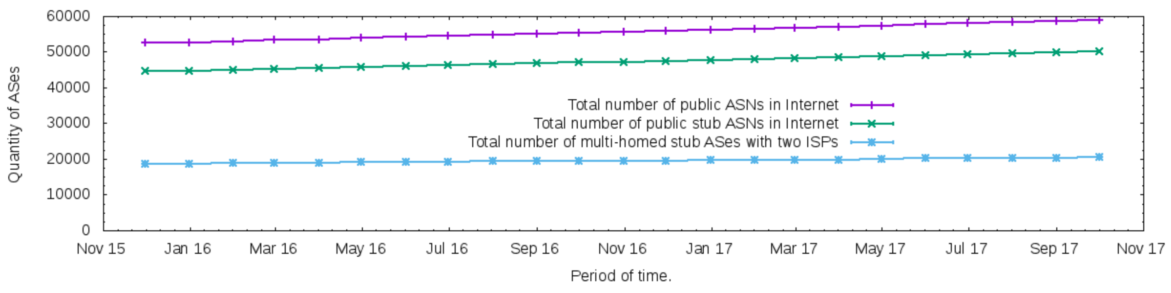
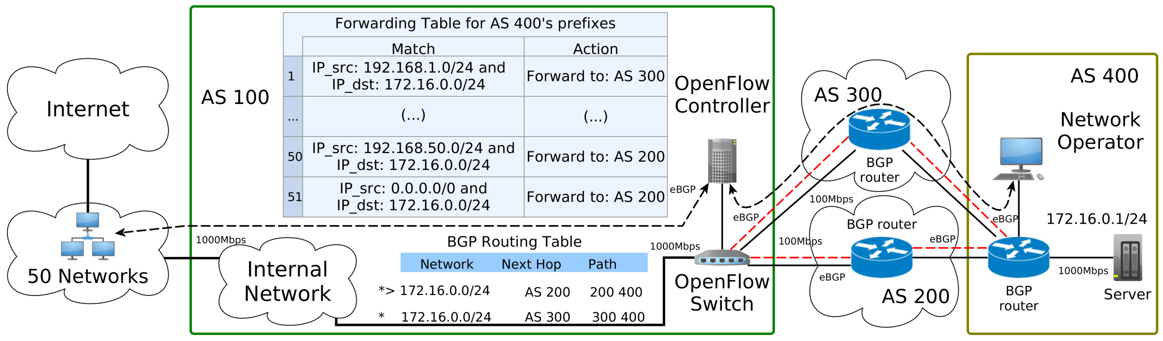


Figure 36 – The total number of multi-homed stub ASs is common in recent months of the Internet.



Forwarding Table for AS 400's prefixes

	Match	Action
1	IP_src: 192.168.1.0/24 and IP_dst: 172.16.0.0/24	Forward to: AS 300
...	(...)	(...)
50	IP_src: 192.168.50.0/24 and IP_dst: 172.16.0.0/24	Forward to: AS 200
51	IP_src: 0.0.0.0/0 and IP_dst: 172.16.0.0/24	Forward to: AS 200

BGP Routing Table

	Network	Next Hop	Path
*>	172.16.0.0/24	AS 200	200 400
*	172.16.0.0/24	AS 300	300 400

Figure 37 – The topology adopted.

The period from December 2015 to October 2017 was considered. Figure 36 presents the total number of public ASN in CAIDA's AS relationship database as well as the total number of stub ASs with public ASN and a subset of this total representing the total number of multi-homed stub ASs with two relationships of customer-to-provider. Thus, as the multi-homed customer stub ASs with two ISP are typical in recent months of the Internet, the evaluation testbed of the proposed architecture adopted a topology with two outbound links, one for each ISP, and a multi-homed customer AS.

Thereby, based on the data above extracted from CAIDA's relationship database, the topology adopted its depiction in the Figure 37. It consisted of four ASs, where three of them were transit ASs (AS 100, AS 300, and AS 200), and one stub AS (AS 400). The relationship of customer-to-provider was: AS 400 to AS 300, AS 400 to AS 200, AS 300 to AS 100, and AS 200 to AS 100. As AS 100 deployed the proposed architecture, a new relationship of customer-to-provider was created between AS 400 and AS 100, with the goal of managing inbound traffic to the customer (in this case, AS 400).

The partial FIB of the AS 100 for the customer AS 400 was indicated in Figure 37, where AS 400 advertised the network prefix 172.16.0.0/24 and the AS 100 learned this prefix and propagated through eBGP sessions between all the ASs in the topology. The AS 100 learned that it could reach network 172.16.0.0/24 either through the AS-path [200,400] or through the AS-path [300,400].

The default behavior of the proposed architecture followed the outcome of the BGP decision process where the preferable next hop to reach the *Server* (IP 172.16.0.1) of AS 400 was by using the path of AS 200. This preferable path is indicated in Figure 37 with the character ">" in the *BGP Routing Table*, and the character "*" indicates that both routes were valid as the result of the BGP decision process. The length of the AS-Path was equal for both paths to the network 172.16.0.0/24. The BGP decision process selected the path [200,400] to be the next hop for that prefix; however, as the *Network Operator* in AS 400 could use the proposed architecture executed in AS 100, AS 400 could explore the use of both paths, if the AS 100 allowed it.

Once there was a relationship between AS 400 (Customer) and AS 100 (Provider) using the proposed architecture, the AS 100 could offload the control of traffic that went to the network prefixes of AS 400. In the topology adopted, only one prefix was available for the customer AS 400 to manipulate, and two options of paths to reach that prefix. Therefore, a load balancer application of AS 400 executing inside the COOL controller in AS 100 could distribute traffic using both of the valid and available paths to reach AS 400. And when only one path becomes available, hence, and the load balancing can not be applied, the traffic will just follow the standard BGP behavior and forwarding all network traffic to just one "best" next hop (per prefix).

5.2.1.2 Prototype environment

To set up the prototype environment, two physical machines were required to implement the topology adopted (see Figure 37). One physical machine executed the following components of the prototype environment: the *OpenFlow Switch*, the *SDN Controller* of AS 100, and the *50 Networks* (number choose by convenience). Each of the *50 Networks* propagated its prefix and had only one virtual machine to generate traffic. For convenience, those prefixes ranged from 192.168.1.0/24 to 192.168.50.0/24. Furthermore, the configuration of the physical machine was a core 8-CPU with a maximum clock rate of 2.20 GHz, 8 GB of RAM, and a network card of 1 Gbps. For the AS 100 configuration, each virtual machine in the *50 Networks* was emulated using a container (lightweight virtualization). For the OpenFlow Switch, the Open vSwitch (OPENVSWITCH, 2016) was used. The protocol enabled in the Open vSwitch was OpenFlow version 1.3. The proposed architecture was executed in the modules of the Ryu SDN controller (RYU, 2016).

The other machine had a 3 GB of RAM, CPU with a maximum clock rate of 2.4 GHz, and a network card of 1 Gbps. This machine was used to create AS 300, AS 200, and AS 400. Those ASs were an instance of the full virtual machines executed in Virtualbox (VIRTUALBOX, 2016) with Quagga (QUAGGA, 2017). The terminal of the virtual machine of AS 400 was used to send and receive JSON messages between the OpenFlow Controller in the AS 100 and the AS 400.

A crossover Ethernet cable was used to connect the physical machines. However, to avoid interference between the OpenFlow Switch and the AS 300 and AS 200, it divided the 1 Gbps bandwidth between the two machines with the Linux *tc* (Traffic Control) tool, allocating two virtual interfaces of 100 Mbps. Thus, the connection between AS 300 and the OpenFlow Switch had an available 100 Mbps as well as the connection between AS 200 and the OpenFlow Switch. Hence, the network traffic from AS 100 to AS 400 is limited to the 100 Mbps passing through path AS 200 and AS 400, as well as the path AS 300 and AS 400.

Regarding the network traffic, the network throughput tests tool were performed with the *iperf* (IPERF, 2016). Each of the virtual machines in *50 Networks* used *iperf* to generate UDP packets in a controllable way. The main motivation for adopting UDP, instead of TCP or other protocol, is the possibility of know how much network traffic is generated in the source and the quantity is actually received in the destination without the congestion control affecting the experiments (see Section 6.2 for future works). Furthermore, for the *Server* component, it was executed in the *iperf* (IPERF, 2016) in UDP server mode, which was started in the AS 400 network. The server was set up with the IP address of 172.16.0.1/24, and connected to the BGP router in the AS 400 infrastructure.

5.2.1.3 Workloads

To test how the management of inbound traffic could be performed using the proposed architecture, several workloads schemes were designed to investigate the bandwidth usage in the prototype environment when applying the three types of strategies. The main goal of the workloads choose is to observe how network traffic can be steering using the proposed architecture in the inter-domain environment. Therefore, the workloads were:

1. **Maximum constant network traffic for BGP behavior**: This workload produces 100 Mbps of bandwidth usage during 30 seconds of using the *iperf* in client mode to generate the UDP packets from the *50 Networks*. The workload was designed to explore the BGP behavior of using just one "best" next hop (per prefix). Thus, based on the adopted topology in Figure 37, this workload will only use one path to carry this traffic, either path [200, 400] or path [300, 400], but not both to reach prefix 172.16.0.0/24 of AS 400.

2. **Constant network traffic with balanced rate (Balanced load)**: This workload produces 200 Mbps of bandwidth usage during 30 seconds of using the *iperf* in client mode to generate the UDP packets from the *50 Networks*. Thereby, the total volume of traffic is split equally into each network. Hence, each one of the *50 Networks* will be responsible for producing 4 Mbps of network traffic using UDP packets during 30 seconds.

3. **Constant network traffic with unbalanced rate (Unbalanced load)**: Similar to constant network traffic with a balanced rate workload, however, with network

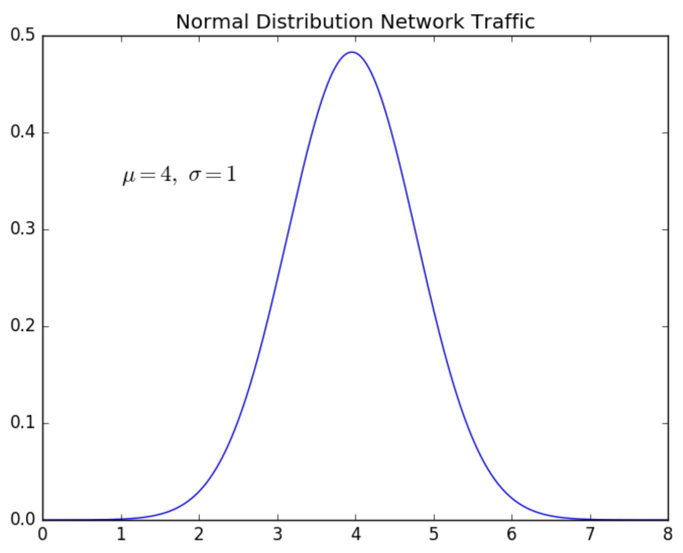


Figure 38 – Expected normal distribution of network traffic for the workload.

traffic configured to become unbalanced. Thus, the *50 Networks* is split in half, and 25 of the *50 Networks* will generate 7 Mbps and the other 25 will generate 1 Mbps of network traffic using UDP packets for 30 seconds, where the total number of bandwidth produced is 200 Mbps in this interval.

4. **Normal distribution workload (Normal distribution)**: this workload produces 200 Mbps of bandwidth using UDP packets following a normal distribution for the *50 Networks*. The mean of the normal distribution is 4 Mbps of bandwidth with a standard deviation of 1 Mbps. The representation of the expected normal distribution of the network traffic for this type of workload is in Figure 38.

5.2.1.4 Results of the workload execution

The experiments were performed to investigate how different strategies could play a role in managing incoming traffic. It used the topology adopted (see Figure 37), and explored the three strategies for controlling inbound traffic: RLL (Random); RRLL (Round-Robin); and, RRLLT (Round-Robin with threshold)[3].

The three types of workload were applied to the prototype environment for the RLL strategy (workloads 2, 3 and 4 were considered). The experiment executed more than 1000 times, the mean and the standard deviation were calculated. Then, the confidence interval with a 95% confidence level was calculated as well, and the Figure 39 presents the results for RLL. The total bandwidth used in the experiments was 200 Mbps and it was expected that the RLL distributed this load for both links of the AS 100. The ideal scenario was

[3] See Section 4.4 for more details about each algorithm.

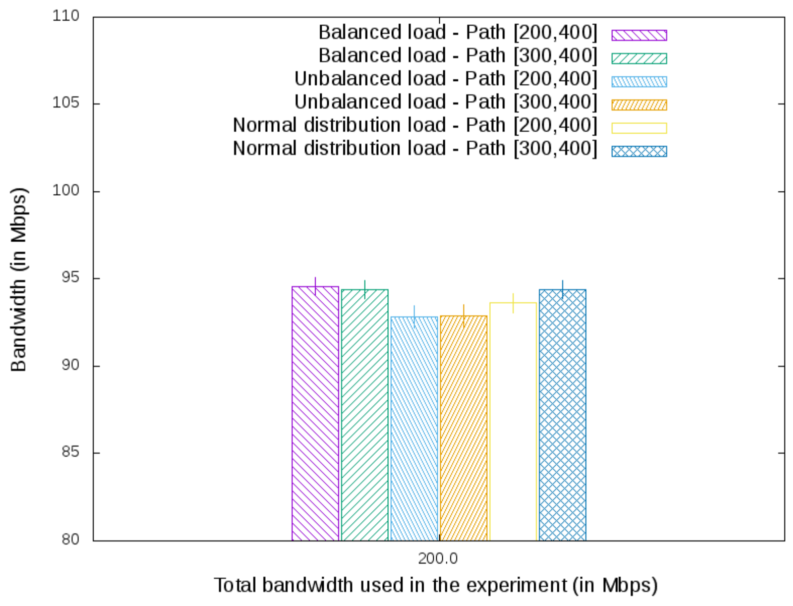


Figure 39 – Results using the Random strategy.

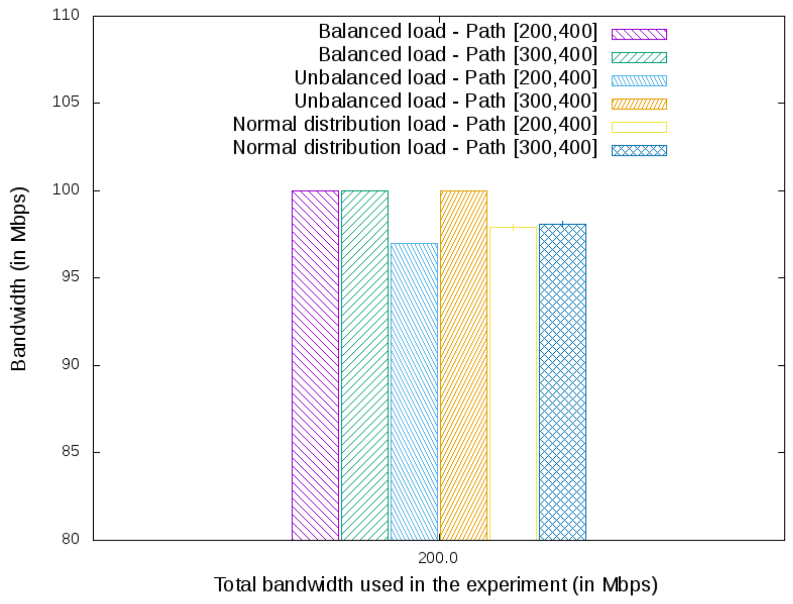


Figure 40 – Results using the Round-Robin strategy.

RLL send 100 Mbps using path [200, 400] and 100 Mbps using path [300, 400] to reach AS 400 (see topology adopted in Figure 37). However, for all types of workload executed, including workloads 2, 3 and 4, none of the paths barely reached 95% of the utilization (95 Mbps of bandwidth usage). Although the *Balanced load* (workload 2) seems to reach better bandwidth usage for both paths, it is wise to claim that it poorly distributed the load between the available links of the AS 100.

For the RRLL (Round-Robin strategy), the workloads 2, 3 and 4, the same process of statistic computation was performed: 1000 samples, mean and standard deviation computed, and the application of the 95% confidence level. The results are in Figure 40. The

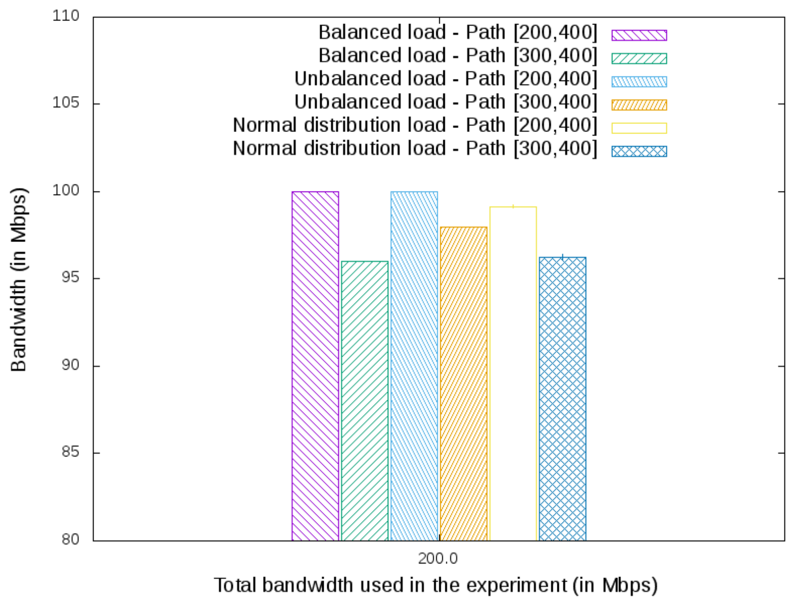


Figure 41 – Results using the Round-Robin with Threshold strategy using 95% of the link capacity for all outbound links.

Balanced load workload provides the best result for the RRLL, reaching the full bandwidth utilization available (200 Mbps). The same results were not reproduced for *Unbalanced load* and *Normal distribution load*, the workloads 3 and 4, respectively. However, the results for RRLL were better that those found with RLL, for every workload used.

Figure 41 presents the results for RRLLT (Round-Robin with threshold). The best next hop for reach the prefix 172.16.0.1/24 in the AS 400 is through path [200, 400] (see the *Forwarding Table of AS 400's prefixes* in Figure 37). Thus, the strategy will make the links usable until 95% of its capacity, when the link capacity overlaps this threshold, then it will be considered congested, and RRLLT will not use the link anymore. When all alternative paths are congested, the default path is used (in the case, the best next hop) even if it is also congested.

For all workloads used (2, 3 and 4), when the path [300, 400] reached the threshold of 95% capacity, the default path [200, 400] was used. Hence, this behavior created a characteristic pattern in Figure 41, where path [300, 400] did not reach the full capacity (because it was considered saturated) and the path [200, 400] did.

Figure 42 presents the aggregation of packet losses during the experiment of managing inter-domain traffic. It considered both paths [200, 400] and [300, 400] simultaneously. Once the UDP does not provide capabilities to provide the quantity packet loss, this metric considered the expected bandwidth (200 Mbps) minus the real quantity received by the *Server* 172.16.0.1. The experiment used the workloads 2, 3 and 4, and it was executed 1000 times. The confidence level of 95% was used.

The evaluation of the packet loss provides information about how the SDN application performed. Analyzing the workload 2, the RRLL did not lose any packet, and it was the

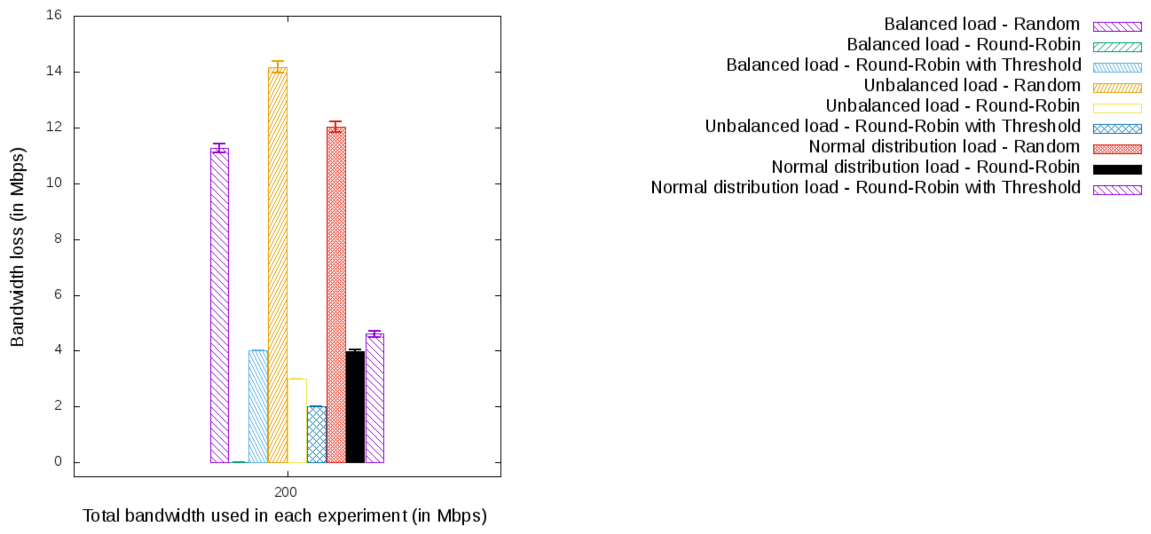


Figure 42 – Packet loss.

only result where the application reached the full capacity of both links of the domain AS 100. Besides, the RRLLT performed better than RLL. For the *Unbalanced load* (workload 3) the RLL presented the more packet loss compared against RRLL and RRLLT, with the RRLLT presenting less packet loss than RRLL. Finally, the workload 4 (normal distribution load) produced equally quantity of packet loss for RRLL and RRLLT, with the confidence intervals overlapping. The RLL produce more packet loss than RRLL and RRLLT for the three workloads adopted.

Based on the results found, the selection of the load balancing strategy depends on the type of workload in the traffic between domains. Network traffic without much variation in the bandwidth usage, such as the workload 2 (behaved network traffic), it was more suitable for reactive load balancing using the Round-Robin strategy as it was used for RRLL. The RRLLT indicates to be at least equally good as the RRLL for unbalanced loads (workloads 3 or 4). And the Random strategy (as the RLL) should be avoided to distribute traffic when an option to adopt a reactive load balancing for SDN application is available.

5.2.2 Protection against external link failure

This scenario captured the dynamic of a disruption of an external link when the domain that deployed the proposed architecture was protected with the FF rules of OpenFlow. The main goal of this scenario was to mitigate the unavailable time of the domain during the disruption of an external link, taking into account the FF mechanism and exploring the multiple paths allowed by the proposed architecture.

This scenario used Workload 1 and the prototype environment describing the adopted topology (see Figure 37). To avoid any delay or packet losses that could be caused during the OpenFlow rules installation process, all FF rules were installed before the execution

Table 12 – Flow table and group table for the *OpenFlow Switch* using OpenFlow Fast Failover Group Table.

Flow table at *OpenFlow Switch.*

Match fields	**Instruction**
IP_src:192.168.1.0/24 and IP_dst:172.16.0.0/24	Group1
IP_src:192.168.2.0/24 and IP_dst:172.16.0.0/24	Group1
(...)	Group1
IP_src:192.168.50.0/24 and IP_dst:172.16.0.0/24	Group1
IP_dst:172.16.0.0/24	Group2

Group table at *OpenFlow Switch.*

Group Identifier	**Group Type**	**Action Buckets**
Group1	Fast Failover	Watch: AS 200 port; Outport: AS 200 Watch: AS 300 port; Outport: AS 300
Group2	Fast Failover	Watch: AS 200 port; Outport: AS 200

of the workload. An example of the state of the table rules is depicted in Table 12.

Suppose that *OpenFlow Switch* of Figure 37 is the edge device of AS 100. As the workload was designed to explore the BGP's destination-based forwarding paradigm, the best next hop for the prefix 172.16.0.0/24 is through the AS 200. If the domain AS 400 can apply inbound control over the prefix 172.16.0.0/24 in the AS 100, then all traffic from the *50 Networks* can be forwarded as the AS 400's interest.

The first 50 lines of the *OpenFlow Switch* flow table reflected this scenario, and used the group table *Group1* identifier, where it forwarded packets to AS 200 when the external link to this AS was live (not failed), or otherwise to AS 300. The last OpenFlow rule is the routing safety rule, where all the remaining traffic that does not match against previous rules will be forwarded. The *Group2* identifier was used to forward network traffic for path [200,400], which was the "best" next hop for the BGP AS 400's prefix.

Regarding the experiments, Figure 43 depicts the average bandwidth received by the *Server* during the execution of Workload 1. The workload was executed more than 30 times with a confidence level of 95%. Figure 43 indicates that the FF mechanism did not affect the total bandwidth available to the workload once the maximum available bandwidth was used practically during the entire 30 s interval.

In fact, a sample of the execution of the selected workload is presented in Figure 44, which shows how the traffic used different paths to reach the *Server*. Although Figure 44 describes the network behavior of the traffic generated during the 30 s, it does not provide details about the disruption event that occurred in the 15th second. Next, Figure 45 details what happened in between the times of 14.95 s and 15.05 s of this scenario.

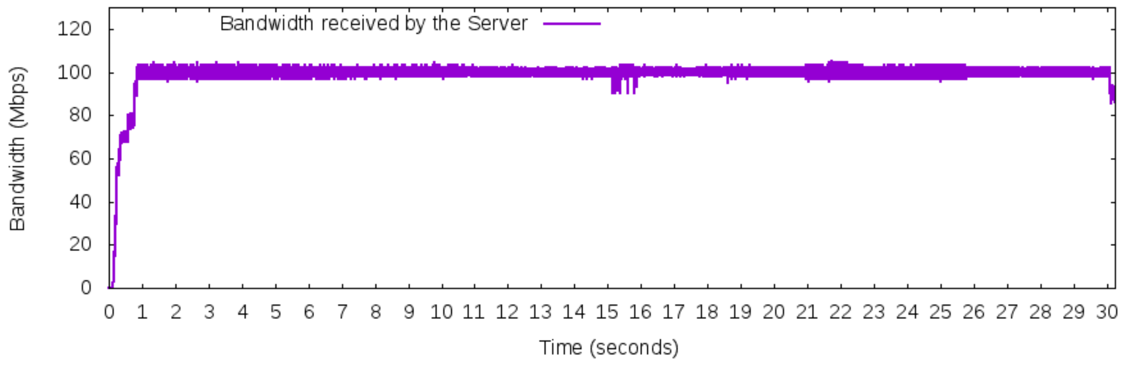


Figure 43 – The average of multiple measurements for the network traffic during a link failure around the 15th second with a confidence level of 95%.

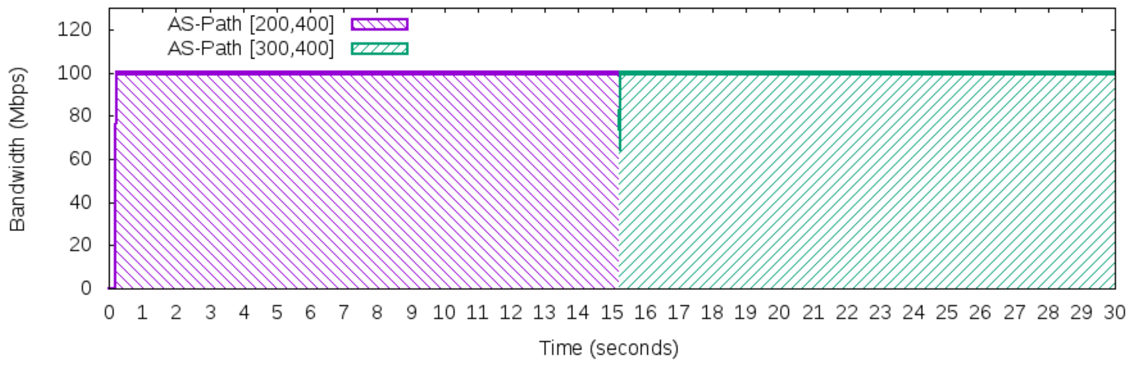


Figure 44 – An execution of link failure in the 15th second.

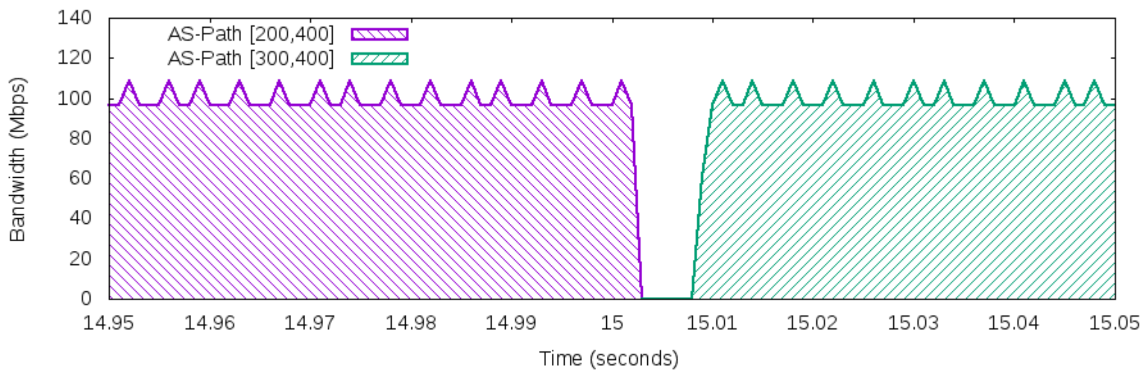


Figure 45 – Details of network traffic for the link failure around the 15th second.

The workload produced 100 Mbps of network traffic and used the path [200, 400] as the "best" next hop and path [300, 400] as a backup path. In the 15th second, the link connecting AS 100 and AS 200 failed. After the link failure in the 15th second, the FF mechanisms took a few milliseconds to discover the disruption. Once this occurred, the traffic automatically shifted from path [200, 400] to path [300, 400]. This process was done exclusively in the data plane (OpenFlow switches) and took less than ten milliseconds, which is an acceptable value for disruption in an inter-domain environment (less than

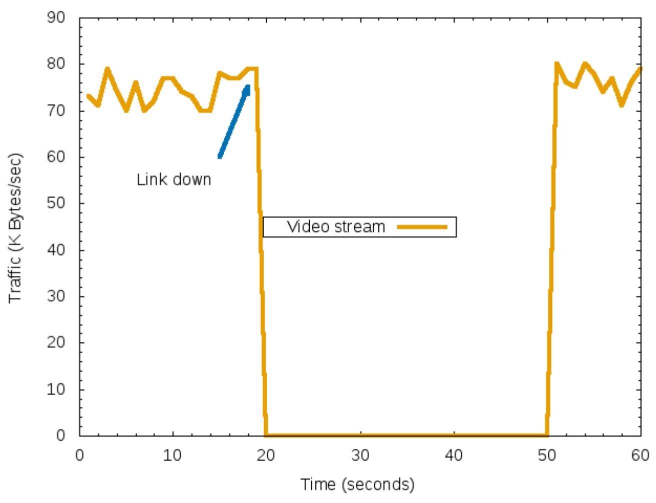


Figure 46 – Link recovery to video stream using SDI solution.

50 ms (Niven-Jenkins, B and Brungard, D and Betts, M and Sprecher, N and Ueno, 2009)). The measured millisecond interval was coherent with the FF mechanism recovery times found in the literature (SHARMA et al., 2011; LIN et al., 2016).

The main result of this scenario was that, by exploring the valid and available multiple paths of BGP, the proposed architecture avoided the application of the BGP decision process for selecting a new best next hop when an external link failed. The BGP convergence is a time consuming process (dozens or hundreds of seconds (WANG et al., 2015; GODFREY et al., 2015)) when deciding a new route, and with the application of the proposed architecture mechanisms, if it exists, a valid next hop will then always be already computed and installed in the data plane, and then the domain can immediately use the spare route. Therefore, the proposed architecture can bring the unavailability of paths to the milliseconds time after an external link failure, instead of being subject to the long BGP convergence time.

Regarding of related works, the Software Defined Inter-domain (SDI) routing (WANG et al., 2016a) advanced into the inter-domain support of flexible routing policies to forwarding packets. Instead of just use the BGP's destination-based forwarding paradigm, SDI adopted update rules with multiple fields of the IP packet header to construct the inter-domain global view of the solution. To show the flexibility of fine-grained routing policies in multi-domain SDN networks, a prototype to experiment SDI was conducted, and a test was done to analyze the behavior of SDI solution when an external link suffers a failure. The results are in Figure 46, and SDI took a few seconds to restore the connectivity. Despite the fact SDI is an SDN solution, let the SDN controller manage the restoration of an external link failure requires the computation of a new network state inside the controller. This process can take a few seconds to reach the proper response.

Although creating flows with a reactive approach can save memory space inside data plane devices (SILVA, 2017), as it is the case of SDI, the most significant shortcoming of those approaches is the restoration time. The FF feature of OpenFlow protocol can be classified as a proactive approach (NGUYEN et al., 2016), where rules are populated in advance. Thus network flows are created before the packet arrives at an OpenFlow switch port. Hence, a flow will be already installed into the switch and the response time is drastically reduced, once the restoration mechanism is very close to where the network problem occurred.

5.2.3 Evaluation of Traffic Exchange Messages

We elaborated a simulation to evaluate the traffic exchange requirements in the inter-domain environment for the proposed architecture. Thus, the simulation program can execute a few algorithms extracted from the literature using SDN technologies (see Chapter 3). For convenience, the Hop-by-hop (CHEN et al., 2016b), MLV (CHEN et al., 2016b), BGP and BGP with the proposed architecture exchange mechanisms was implemented [4].

Unfortunately, the authors of Hop-by-hop and MLV were unable to provide the actual software used in their work, so we developed the algorithms based on their paper (CHEN et al., 2016b). Hop-by-hop and MLV uses a link-vector data structure to exchange the network state, and MLV was presented in Chapter 3. The BGP and BGP with the proposed architecture use a path-vector data structure.

We modeled the traffic exchange of BGP as in a process of Breadth First Search (BFS) algorithm, as Chen et. al. (CHEN et al., 2016a) indicated. To BGP with the proposed architecture, we considered that once the ISP discover a multipath to a customer, it will send a message to the customer with the size of a prefix.

The topology adopted for executing the simulations is in Figure 47. It will be the baseline for the simulation in this Subsection. Although this topology is a hierarchical structure (GAO, 2001; CHEN et al., 2016b) to represent the Internet, there are works in the literature that put this claim in perspective (e.g., Luckie et al. (LUCKIE et al., 2013)). However, we choose this type of topology to enable the comparison between related works and the proposed architecture.

Figure 47 represents the ASs in circles, and the number indicates its ASN. Furthermore, each connection represents the relationship between ASs. If the relationship is *Customer to Provider*, then the provider receives prefixes from the customer and advertises to others ASs. And in the *Peer to Peer* relationship, both ASs exchange their learned prefixes.

MLV's work used 4 bytes for the prefix field and 32 bytes for the link field (CHEN et al., 2016b), and did not considered any overhead caused by the carrying protocol, such as the TCP is to BGP. Thereby, the simulation with BGP and BGP with the proposed architecture will not incorporate the overhead caused by the transport protocol as well.

[4] The simulation program is in the URL: <https://github.com/walberjose/Simulator>

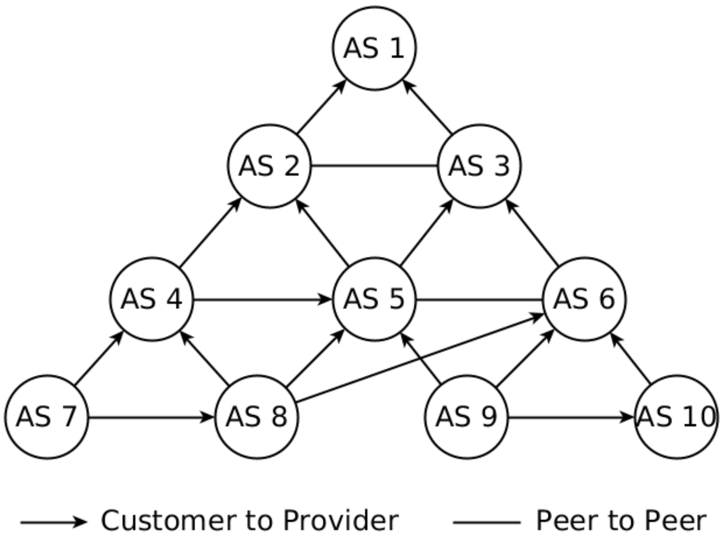


Figure 47 – Topology used to evaluate MLV mechanism, and adopted in this work for the exchange traffic simulation.

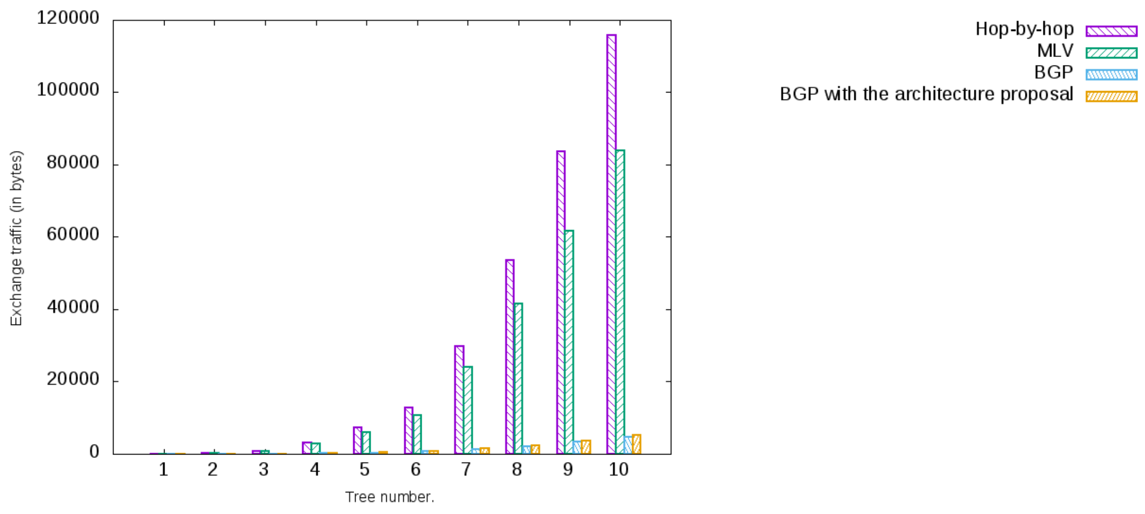


Figure 48 – Exchange traffic by tree number.

Figure 48 represents the growing of exchange traffic for each AS that advertises only one prefix and increases the number of ASs in the topology of Figure 47 for each iteration. Thus, as more ASs are incorporated into the topology, the quantity of exchange traffic increases to reach the steady-state of the network. As notable, the strategies of Hop-by-hop and MLV are more sensitive to the increase of the topology (new nodes added) than BGP and BGP with the architecture proposal. The BGP with the architecture proposal consumes more traffic than only using BGP in the experiments. This outcome occurs due to the need to send messages back to the customers when using BGP with the architecture proposal.

In Figure 49, the number of prefixes advertised per AS is investigated. This graph considered all the nodes in the topology and varied the number of prefixes advertised for

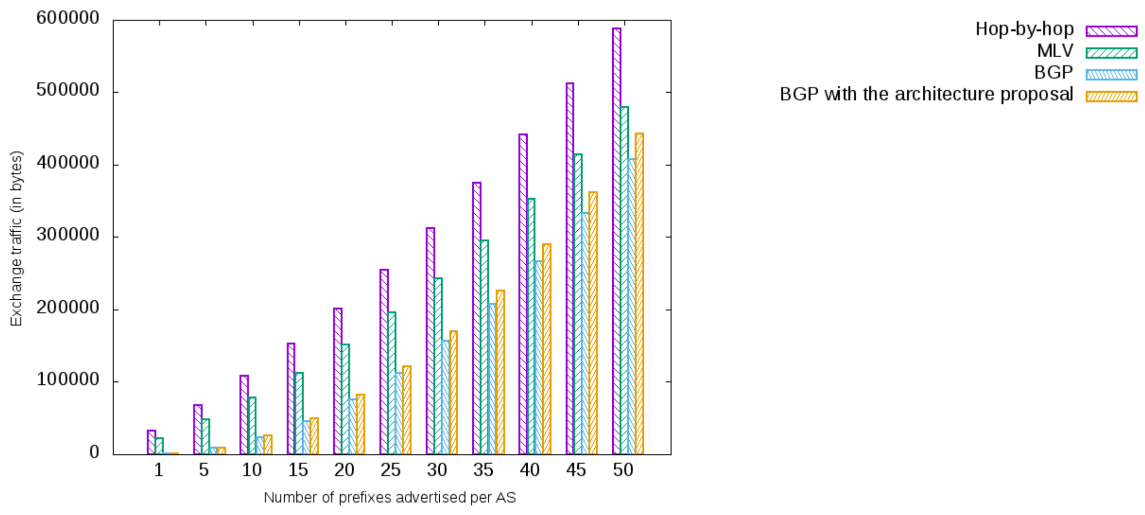


Figure 49 – Exchange traffic.

each AS in the topology of Figure 47 from 1 to 50 with increment of 5 prefixes per AS. Analyzing the results of Figure 47 and Figure 48, there is an evidence that the number of ASs in the topology affects more the traffic exchange than the increase in advertised prefixes per AS.

Another observation is that in Figure 49 as the number of prefixes advertised per AS increases the BGP and the BGP with the architecture proposal starting to get close for Hop-by-hop and MLV solution in terms of exchange traffic. However, for the inter-domain environment is better to have solutions less sensitive to the increase of nodes than the number of prefixes advertised. Today, the number of public prefixes on the Internet is less than a million (CAIDA, 2017), although the number of routers (as a simplification of the number of nodes) is unknown, but it is expected that this value is to be around of 1 billion.

5.3 ARCHITECTURE DISCUSSION

Although the main results found in this Thesis are in the last section, here presents some complementary discussion about the proposed architecture.

5.3.1 Discussion based on the related works

BGP follows the destination-based forwarding paradigm (REKHTER; LI; HARES, 2006) and uses one "best" route per prefix. Although these features enabled the scalability of the BGP protocol, they provide a limitation for the inter-domain routing system to explore the path diversity on the Internet (QIN et al., 2012). Thereby, the baseline for forwarding traffic is based on the BGP's gross-grain routing policies.

SDN enabled new approaches to improve the inter-domain ecosystem, and because the routing and policies tasks are embedded into the BGP protocol, new solutions adopting SDN technologies try to overcome the BGP limitations. For example, the effects of SDN centralization on Internet routing convergence was studied in recent works (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015; CHEN et al., 2016a; GÄMPERLI; KOTRONIS; DIMITROPOULOS, 2014), where the simulation of those works indicates the effectiveness of SDN utilization in the inter-domain routing environment.

Thai and Oliveira (THAI; De Oliveira, 2013) presented the Interdomain Management Layer (IML) architecture to allow SDN networks to exchange network resources. The goal of IML is decoupling policy from routing for facilitating inter-domain resource sharing between ASs. In the IML framework, it did not have detailed how fine-grain inter-domain routing policies rules could be achieved.

Kotronis et al. (KOTRONIS; GAMPERLI; DIMITROPOULOS, 2015) developed a hybrid BGP-SDN emulator framework for evaluating for the transition between traditional routing technologies (e.g., BGP) to SDN, called SIREN. Although SIREN is compatible with multiple traditional protocols, it adopted the BGP for exchanging control information between SDN domains, where some works indicated that BGP is a weak candidate to promote the next generation of SDN's exchange network control, once the control of the protocol has a limited set of capabilities (THAI; De Oliveira, 2013). Thus, this Thesis proposed an architecture that explores the BGP capabilities and uses SDN technologies, with a new control channel included, to enable routing applications to define new logics to the inter-domain routing system.

The Software Defined Inter-domain (SDI) routing (WANG et al., 2016a) advanced into the gap of supporting inter-domain flexible routing policies to forwarding packets. SDI used WE-Bridge to exchange control messages between SDI domains. And the Multi-dimension Link Vector (MLV) (CHEN et al., 2016b) depicted a new mechanism to exchange the network view using a link vector algorithm (BGP uses a path vector algorithm). Thereby, the cumbersome of SDI and MLV solutions is that they propagate network link states into the inter-domain routing system and do not have backward compatibility with the current Internet infrastructure.

5.3.2 Discussion about the partial deployment of the proposed architecture

The BGP is well known an “ossified” protocol (SILVA; Djamel Fawzi Hadj Sadok, 2018) and the last decades demonstrated that previous proposals to overcome the limitations of that protocol are hard to achieve. Thus, SDN proposals incompatible with BGP have limited appeal for real-world deployments (SILVA; Djamel Fawzi Hadj Sadok, 2018), e.g., MLV (CHEN et al., 2016b). Therefore, the proposed mechanism provides fine-grain inter-domain routing policies using the data structures extracted from BGP (guarantee the backward compatibility) and uses inter-domain communication interface to exchange control information between domains to apply those new types of routing policies. Being backward compatible with BGP is an incentive to the adoption of the proposed mechanism for ISPs with SDN

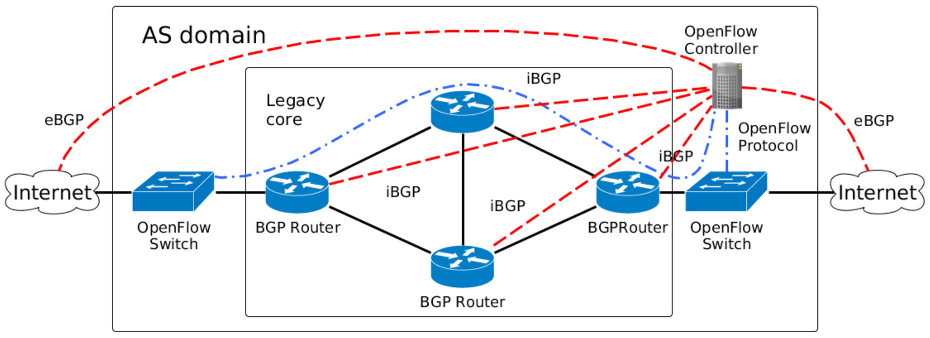


Figure 50 – Scheme for backward compatibility with legacy network.

networks.

Therefore, this subsection tackles the argument about partial deployment. We split this analysis into two fronts. In the first part, incremental deployment of the proposed architecture inside the AS and the other part is about how we think inter-domain should evolve by adopting the proposed architecture.

5.3.2.1 Backward compatibility with legacy networks internally

A well-established design principle is the end-to-end argument (SALTZER; REED; CLARK, 1984). Based on this principle, the network complexity has to be concentrated in the network edges, making the core simple from the perspective of network management. Thus, the proposed architecture applies this design principle, and the programmable devices (notably OpenFlow switches) are positioned at the edges of the network. Hence, two options for the core might occur:

- **Legacy network**: With the legacy network in the core, the SDN controller is the centralization logic that acts as *Route Server* for the BGP routers. Figure 50 depicts a scheme for mixing OpenFlow switches and legacy network appliances. The legacy core is connected through internal BGP (iBGP) sessions to the OpenFlow Controller, which distributes external routes to the internal BGP routers. The infrastructure routes continue to be distributed with the interior gateway protocols (such as OSPF) executed in the *Legacy core* perimeter.

- **OpenFlow network**: In the future, it is expected that all network devices will have some SDN technology embedded inside an AS. Thus, a domain with SDN technology (OpenFlow devices) is straightforward, and Figure 51 depicts this scenario. All OpenFlow switches are connected to the OpenFlow Controller, which manages the flows inside the AS domain.

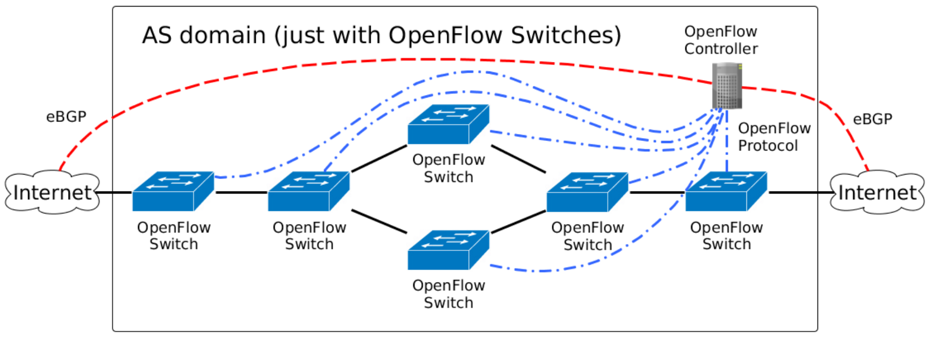


Figure 51 – AS domain composed of only OpenFlow switches.

5.3.2.2 Backward compatibility with legacy networks in the inter-domain

Once not all of the AS will adopt the proposed architecture (at least not right away), it is mandatory to investigate how the incremental deployment should be rolled out in the inter-domain environment. To investigate the potential of the proposed architecture on the Internet, CAIDA's database (CAIDA, 2017) was used to rank the ASs that provide transit to other ASs. The ranking was based on the *Customer Cone* metric[5], which measures the number of direct and indirect customers of a given AS. The data for ranking the ASs was based on the *Customer Cone* metric and was extracted from the topological data collected by CAIDA and BGP routing in June 2016. A total of 54,772 ASs comprised the dataset used.

The first ten ASs in this ranking were selected and the number of ASs in their *Customer Cone* were measured. Figure 52 depicts the potential of the proposed architecture in the Internet. The first 10 ASs in the ranking had an expressive quantity of direct or indirect customer relationships that could allow those customers to use the benefits of the deployment of the proposed architecture. Therefore, if one of those 10 ASs provided network services to control the inbound management of network traffic to its customer in the cone, those customer ASs could explore the path diversity of the Internet by using an *AS app* executing inside the provider.

The proposed architecture acted in the BGP decision process and allows an AS that deployed this technology to offload its control for customer ASs. Next, a customer could manage inbound traffic by applying control mechanisms for network traffic toward its prefixes. The premise was that ASs would be willing to have that control and eventually could establish new relationships to transit ASs on the Internet. This type of relationship, where any AS can create a relationship with any other AS, is very difficult in the current Internet architecture.

The benefits for the customers' ASs is the appropriate management of their inbound network traffic and for the transit ASs to provide new network value-added services.

[5] Defined as the set of ASs an AS can reach using customer links (LUCKIE et al., 2013).

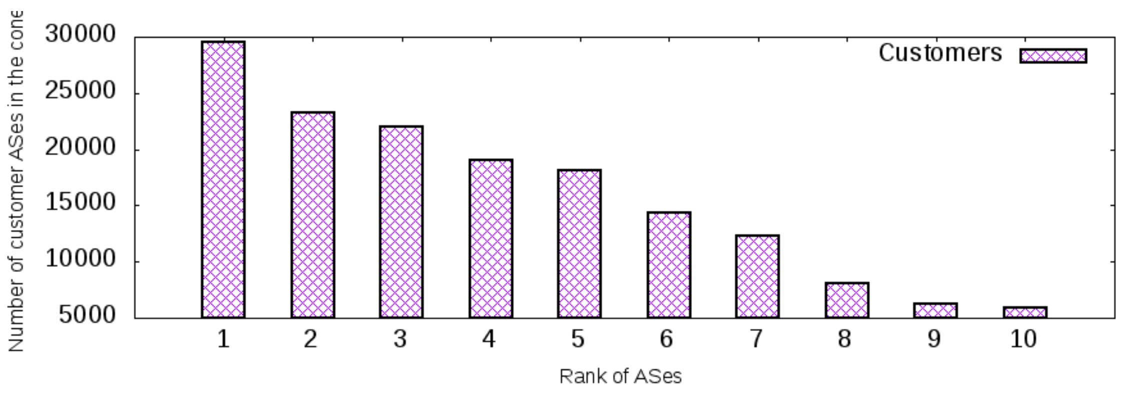


Figure 52 – The potential number of customers affected by the proposed architecture deployed on the Internet.

Obviously, the offload of control from the transit AS would hardly be available for free and will probably provide those new network services if and only if it had some political or financial incentives. Thereby, a transit AS that executes the proposed architecture could explore the new routing services economically and hence increase its profit that would be reachable via collaboration among ASs or due payment. Therefore, a new economy plane for the Internet (WOLF et al., 2014) could emerge and the financial appeal of the proposed architecture is the incentive to transform and evolve the inter-domain environment (SILVA; SADOK, 2017).

Besides having more control of the inbound network traffic, increasing control in the inter-domain routing could also be beneficial for the security of the Internet. For example, it is well known that the mitigation of DDoS is more effective when applying defense mechanisms as close as possible to the source of the attack (ZARGAR; JOSHI; TIPPER, 2013). As the detection is more precise in the AS suffering a DDoS attack, it can respond by expanding its defense mechanism through other ASs. This will have the potential to inhibit the occurrence of DDoS once the defense mechanism can be expanded to being as close as possible to the source of the attack. Future works will focus on mitigating DDoS flooding attacks with the proposed architecture.

5.3.2.3 Logically centralized

Once the OpenFlow Controller is responsible for being aware of the network state, it is important the controller does not become the Single Point of Failure (SPOF) (FONSECA; MOTA, 2017), where if it fails, the entire network will stop to work properly. Thereby, the idea is to have a logically centralized controller, in which there are multiple controllers connected, each one responsible for part of the network and sharing the network state of the network.

The Figure 53 depicts an approach to logically centralized the controller. Each Open-Flow Controller is connected to one or more than one switches via OpenFlow protocol

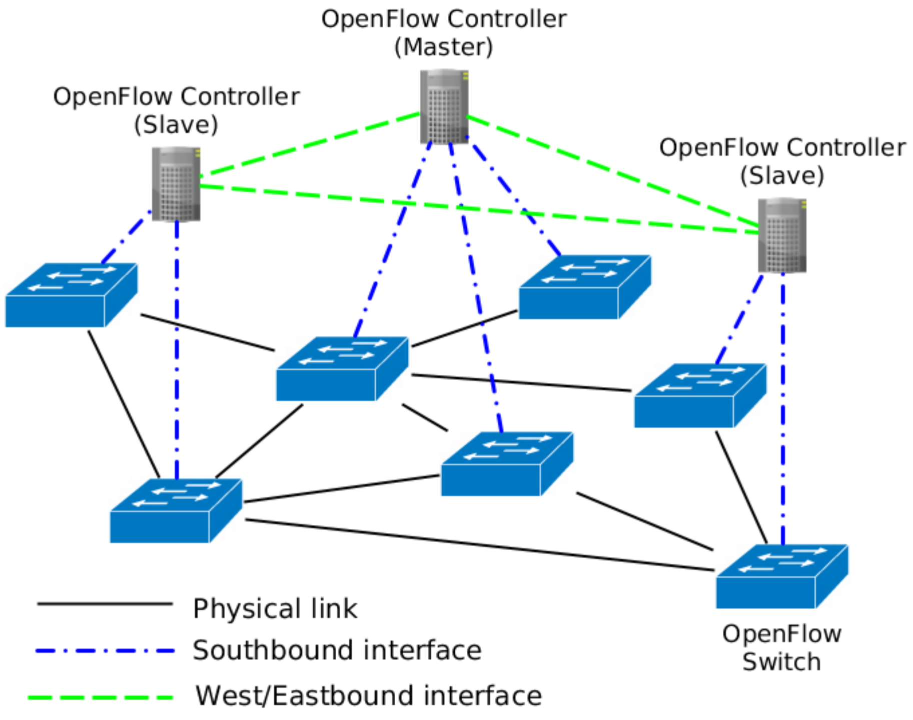

OpenFlow Controller (Master)
OpenFlow Controller (Slave)
OpenFlow Controller (Slave)
OpenFlow Switch
Physical link
Southbound interface
West/Eastbound interface

Figure 53 – Multiple OpenFlow Controllers in a logical centralized scheme.

(the Southbound interface). Also, the controllers connecting with each other using a West-/Eastbound interface. With that connection, the controllers share TE information and often establishes a hierarchic to create the logically centralized network state.

Furthermore, in Figure 53 a Master/Slave scheme is presented, where the Master receives network state from the OpenFlow Controller Slaves to maintain the centralized logic of the network. If a Slave controller becomes unable, the Master controller takes control of the OpenFlow Switches controlled by the unavailable controller. In the case of the Master Controller being down, an election should occur to elect a new Master controller for the network.

5.3.3 Adopting a Routing Registry

One major concern in BGP inter-domain routing is the exploration and impersonation of the ownership of IP prefixes for ASs. Hijacking, misconfiguration, and errors (ALSHAMRANI; GHITA, 2016) can lead to the disconnection of portions of the Internet (e.g., creating black hole routing), MITM, DDoS, and other security issues. These issues occur because the control plane of BGP does not provide security mechanisms for discovering what an AS owns and is allowed to advertise for a given prefix.

To be reachable on the Internet, an AS has to inform the prefixes from its network and passes those values to its ISP (or its Peers). Often, this is done basically by signed contracts between ASs during the establishment of their AS relationships, and an agreement for which RR is used. An Internet Routing Registry (IRR) is a database on ASNs and IP routing prefixes. A registry in an IRR is used for configuring BGP routers to avoid security issues between ASs (IRR, 2017).

When an RR is adopted, it is possible to build filters based on the registries of that database. Although there are some global IRRs (for example, ARIN's IRR (IRR, 2017)), often the providers run their *Routing Registries*. The problem of using a global *Internet Routing Registry* is that the current efforts do not provide extensive, updated, and reliable databases of most of the ASs, and hence the registries usually do not reflect the reality of the ASs relationships and the ownership of their prefixes.

However, a private RR for an ISP that deploys the proposed architecture can provide a properly level of trust, once all the AS agrees about the integrity of the registries. Thereby, when a customer ASs request some modification about the path for its prefixes, the provider can verify that those prefixes belong to the customer and then apply those actions.

5.3.4 Controlling inbound network traffic

The Internet is intrinsically heterogeneous and distributed where network operators execute independent network management tasks into their AS domain (LIN et al., 2014). Additionally, the lack of direct BGP mechanisms to control inbound network traffic contributes to the over-provisioning of ASs infrastructure, leading to an inter-domain environment consisting of expensive, high-performance, and specialized hardware. Consequently, these network resources are underutilized by around 30% to 60% (HONG et al., 2013) to cope with disruption or unpredictable changes of connectivity in the inter-domain environment.

Moreover, the BGP does not provide an effective mechanism to control inbound traffic for prefixes advertised to other ASs. All traffic engineering tools available with BGP only rely on techniques that try to influence other BGP speakers to generate the network traffic distribution desired. Thus, this work explored the possibility of adopting SDN technologies in the inter-domain environment (for example, an ISP) and providing new routing network services to customers of the transit ASs that deploy the proposed architecture.

A prototype was used to execute an experiment of how to perform load balancing for inbound traffic to a given AS. Although the scenario adopted consisted of few ASs, a common configuration in current Internet topology was extracted, which was expected to be represented in a real scenario deployment. Furthermore, based on the achieved results, the proposed architecture was effective for allowing customers of a transit AS to control the inbound traffic.

5.3.5 Discussion about inter-domain routing

Suppose the composition presented in the Figure 54. The stub AS C is connected to the Internet using two links and advertise a route prefix 10.0.0.0/24 to all its neighbors. Furthermore, AS C has some interest in steer network traffic originated in AS A and AS B based on the origin of the traffic. The desire of AS C is to receiving traffic to prefix 10.0.0.0/24 from AS A through path [AS A, ISP X, Path 1, AS C] and from AS B through path [AS B, ISP X, Path 2, AS C].

The BGP provides techniques to *influence* the incoming network traffic towards its network prefix (CARDONA et al., 2016), for example, *AS path prepending*, *Multi-Exit Discriminator (MED)*, *Selective announcements*, and *Community*. In the case of Figure 54, the ISP X will learn the prefix 10.0.0.0/24 from Link 1 and 2. Because of the longest prefix match is equal for those prefixes, the BGP will apply the BGP decision process to decide the "best" path to that prefix. Using the BGP path selection decision process (see Table 1), the MED and *AS path* only will be used after the *Local Preference* value is verified. Thus, altering the value of MED and AS-path (using the *AS path prepending* technique), in the BGP update message, does not provide guarantees for the AS C that the traffic will be forwarding as its interest.

Besides, splitting the prefix 10.0.0.0/24 and advertising small prefixes through its inter-domain links (thus, applying *Selective announcement* technique) could produce a result next to the interest of AS C. However, it is very unlikely that splitting a "/24" prefix will be accepted by its neighbors or others ASs in the Internet, because filtering small prefixes (e.g., "/25", "/30", etc.) is the best practice in the ISP community (SMITH, 2017).

Shao et al. (SHAO et al., 2015b) depicted multiple scenarios where the *BGP Communities* could be used to provide some improvement on the "fine-grain" inbound traffic engineering with BGP. The idea of *Community* is to mark a piece of information on the prefix advertised by the AS that originated it (such as AS C in Figure 54), and use that information in the foreign AS (e.g., ISP X). However, the authors indicated that although some scenarios a given AS can achieve the proper network traffic distribution towards its domain, others complex scenarios (e.g., the case of Figure 54) cannot be resolved using BGP, even whether ISP X is willing to cooperate with AS C. Therefore, a network operator that applies the BGP traffic engineering mechanisms will not achieve a network traffic distribution that satisfies its inter-domain routing polices interest.

5.3.5.1 Inter-domain routing policies conflicts

Because the ASs are autonomous to decide its own TE policies, the conflict (or tension) between inbound routing policies of the tail-end AS against the outbound routing policies of ISP is inevitable. Therefore, some options for conflict resolution of routing policies between different ASs are:

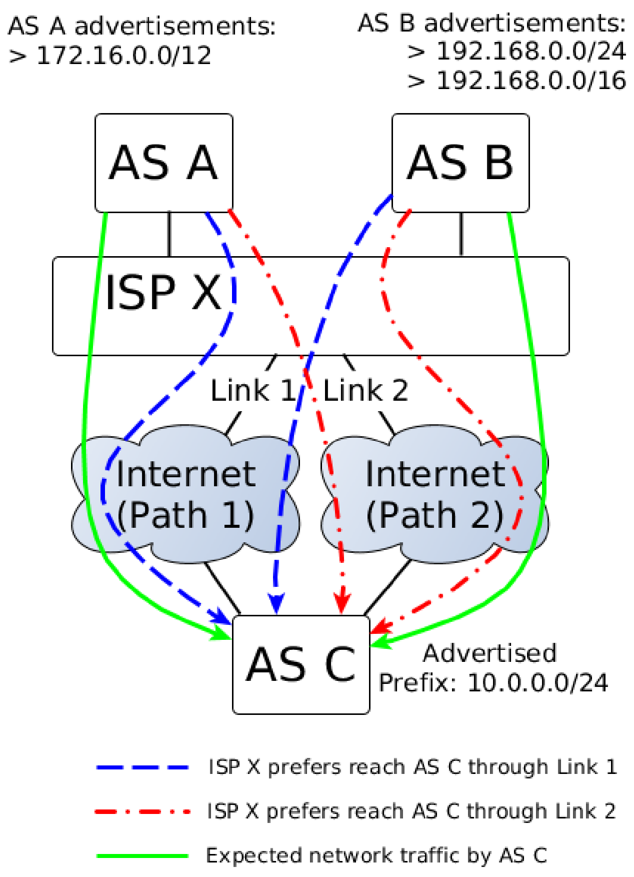


Figure 54 – Using standard BGP implementation, AS C can only influence the routing policies of ISP X to forward network traffic towards AS C that will use either Link 1 or Link 2, but not both at the same time.

- Prioritize directly connected ISP's clients: the ISP that deploys the proposed mechanism allows direct connected. In Figure 54, this conflict resolution occurs when the ISP X gives priority for AS A and AS B instead of AS C.
- Prioritize ASs that originated prefixes: once the ISP has information about the origin of a prefix (e.g., using an IRR database), it can provide priority to ASs that originated the prefix over others ASs. For example, in Figure 54, AS C will allow dictating its preference about the network traffic path over AS A or AS B.
- Prioritize a third party to control (steering) traffic inside the ISP: In this scenario the client does not own the prefixes addresses for the origin or the destination of the network packets.

Thus, the proposed architecture does not impose, or restrict, what ISPs should do or not do for its inter-domain routing policies, but instead, it overcomes the limitations of BGP protocol and give more freedom to the ISPs in defining its own fine-grained inter-domain routing policies and its clients.

5.3.5.2 Bilateral agreements

The proposed mechanism was designed to bilateral incentive agreements between different domains on the Internet. The idea is to establish connections between domains that require applying fine-grained inter-domain routing policies. Thus, by using the direct Inter-domain communication interface, the ISP that executes the proposed mechanism can allow the application of the fine-grain policies and, hence, will mitigate the explosion of propagating, for all inter-domain elements, more specific network states as it is the case of, e.g., MLV and SDI solutions.

In addition to, it the adoption of bilateral agreements also promotes the mitigation of the number of rules to be implemented in the SDN infrastructure for the new types of routing policies produced by the proposed architecture. As highlighted in the previous works (WANG et al., 2016a; CHEN et al., 2016b), propagating the routing control information across multiple domains imposes scalability limitations. As discussed in Subsection 5.1.3.1, the number of network rules in inter-domain environment is proportional to the number of management prefixes in use by the customers of the proposed architecture. Then, instead of the number of prefixes advertised on the Internet become the bottleneck in the inter-domain routing system, the proposed architecture increases the network rules only in ASs that have it installed, and, therefore, bilateral agreements enables the solution to scale.

5.4 CONCLUDING REMARKS

This Chapter executed evaluations on the COOL controller and the proposed architecture. Multiple types of performance evaluation were used: simulation, numerical analysis, and emulation. Each metric adopted verifies a capability or a function. For evaluation of COOL, it was considered resilience mechanisms for OpenFlow networks (including the number of flow entries, signaling overhead, failure recovery time), and a mechanism to avoid inconsistency in flow creation for stateful applications. For the proposed architecture, the management of inter-domain traffic (using different strategies types), the protection of external link failure and the exchange traffic were investigated as well.

Besides, this Chapter discussed the results and implications of this Thesis. The management of network rules is mandatory to properly operate an SDN network in the inter-domain environment. Thereby, the discussion about the number of networks rules in inter-domain environment, the rules installation for OpenFlow networks and a discussion about avoiding inconsistency in SDN stateful applications was done. It is also examined the proposed architecture of this Thesis, and that included a discussion against the related works, the partial deployment of the proposed architecture in the inter-domain environment, the adoption of Routing Registry to increase the security of the architecture operation, the control of inbound network traffic and a discussion about inter-domain routing.

6 FINAL CONSIDERATIONS

In this Thesis, we presented an architecture to evolve the inter-domain routing using SDN technologies, and considered the challenges involved in this network environment. This last chapter delineates limitations (Section 6.1) and a few future works (Section 6.2) and presents the final considerations of this Doctoral Thesis (described in Section 6.3).

6.1 LIMITATIONS

One obvious drawback of the proposed architecture is the adoption of a unique *BGP Speaker* entity executing for the SDN controller (RYU, 2016). This choice made this component a SPOF inside the architecture. And it is well known that SPOF imposes a limitation for scalability since only one instance can establish TCP connections with the others BGP Peers (in practice, the number of TCP ports available for connection is less than 65535). However, the simplicity of using only one *BGP Speaker* was selected by convenience. This limitation can be overcome by creating a new component called, for example, *BGP Speaker management*, with the goal of creating a logically centralized entity that manages the BGP information received and sent to a given domain. Future works can add that network capability for the COOL controller.

Another limitation regards the application of fine-grained rules. For example, it is perfectly acceptable on the Internet that a given prefix be advertised from different ASs, and this case is denoted as Multiple-Origin AS (MOAS) prefixes (JACQUEMART; URVOY-KELLER; BIERSACK, 2014). The current state of this work assumes that each prefix is unique and belongs to only one AS, and that does not reflect the behavior of prefixes advertisement in the Internet.

Although having limitations, this work also opens multiple fronts of research to be investigated. Indeed, multiple academic works referenced the papers produced as the outcome of this doctorate, which is an indication of relevance of this Thesis in the academic field of network computing. Furthermore, the next section depicts some ideas for future works.

6.2 FUTURE WORKS

All the network logic in the SDN architecture resides inside applications executed over SDN controllers. The three strategies for load balancers used in this work followed the reactive flow creation. Hence, for future works, it is interesting to investigate the different approaches for the manipulation of flows in an OpenFlow network, for example, proactive and active strategies for inter-domain environment (SILVA, 2017). To intra-domain, another idea in this line of thought is to manage and apply a more aware congestion control for the flows, e.g., a switchover mechanism (LIN et al., 2016) that changes congested flows to different paths during the lifetime of the flows.

For future works, the execution of different protocols to compose the workload has been considered. One limitation of this work is that it only used UDP flows. The exclusive use of this protocol in the experiments was justified for the sake of space and time. However, the use of the *iperf* tool to set up and generate TCP flows can be considered the next step. TCP flows will be designed to be as representative as possible of the typical Internet TCP flows. The importance of this investigation is to discover the sensitivity of the throughput of TCP flows to dropped packets as realistic Internet traffic tends to be highly dynamic, with individual flows varying in their bandwidth utilization. Thus, with more realistic traffic, it will be possible to analyze whether there is any improvement in the throughput, as was seen in the described experiments using UDP flows.

Another idea to be explored is the creation of a numerical network model to be executed over a BGP topology. A good question to be answered will be how much path diversity on the Internet can be improved with a large adoption of the proposed architecture. IXPs and CDNs can contribute to increasing the capillarity of the Internet traffic, which is reflected in the BGP topology, and the proposed architecture can make it interesting for transit ASs to provide a new type of network services to stub ASs, principally for the management of inbound network traffic.

Besides, we also delineate the following specific points that can complement the work presented in this Doctoral Thesis:

- Create a more refined algorithm for flow creation using different metrics to calculate the paths inside the OpenFlow (OF) network (such as delay, financial cost of links, or any combination of metrics);
- Adopting more representative network traffic distribution to investigate the behavior of the different strategies types for load balancing in the inter-domain environment.
- Modeling the relationships and the link capacity of ASs of the current inter-domain environment to study how much economic gains can be achieved with architecture;
- It is well known that the mitigation of DDoS is more effective when applying defense mechanisms as close as possible to the source of the attack. As the detection is more precise in the AS suffering a DDoS attack, it can respond by expanding its defense mechanism through other ASs, if the proposed architecture is applied on the Internet. This will have the potential to inhibit the occurrence of DDoS once the defense mechanism can be expanded to being as close as possible to the source of the attack. A future work could be applying the proposed architecture on mitigation techniques to avoid DDoS flooding attacks.

- Create a model and simulate the financial gains of an automatic solution for TE using the proposed architecture and the manual configuration. For example, performing inter-domain traffic shift when delays occurs in routes;
- Investigating the use of programmable data plane elements (such as P4) to enhance the control plane capabilities. One interesting feature is to deploy new technologies in the data plane that overcome monitoring limitation of some protocols, e.g. Simple Network Management Protocol (SNMP). One example of new technologies for improving network monitoring capabilities is the gRPC Network Management Interface (gNMI)[1];
- Increment the formality of routing policies, for example, creating a compiler to verify valid policies in the *Domain Controller* component (more specific *Policy Management*). In the current version of the software, the verification occurs only in the manual mode;
- Compare the performance of the applications in the COOL controller against other controllers' applications. Example of SDN applications to be investigated are: path diversity; automated actions (based on link utilization or delay stats); programmable cost function;
- Investigate the inter-controller control traffic of the proposed architecture and others controllers, such as the ONOS controller (MUQADDAS et al., 2016). The idea is to investigate the trade-off between control awareness and overhead communication to achieve it;
- Future works will use more representative network traffic distribution to investigate the behavior of the different strategies types for load balancing in the inter-domain environment;
- As mentioned in Subsection 4.4.1.3, new criteria can be chosen to change OpenFlow Applications for bandwidth control. One more complex algorithm could be applying Feedback Control Theory (FCT) (CHIANG et al., 2007) to continuously control the operation of the routing network (which can be classified as a dynamic system by the FCT). Thus, the error in the FCT would be the quantity between the current bandwidth utilization to the threshold, and the controller will monitor the network to guide the bandwidth utilization for the desired value.

It is expected that those points will be developed and, hence, produce academic articles for future publication.

[1] Publicly available on: <https://github.com/openconfig/reference/tree/master/rpc/gnmi>

6.3 CONCLUSION

This Thesis proposed an architecture to evolve the inter-domain routing using SDN to enable new ways to perform traffic engineering tasks and to mitigate the challenges of using SDN technologies in the inter-domain. Providing a backward compatibility with BGP and using SDN, the architecture allowed a coordination of ASs to perform control task in the inter-domain routing environment.

The proposed architecture also was designed and developed to execute experiments on how the proposed architecture could be used to provide more network traffic control using the capabilities of OpenFlow networks. The results indicated that the effectively of the inbound control mechanisms to manage incoming traffic for the inter-domain routing was dependent on the logic executed in applications of the proposed architecture.

To the SDN challenges of managing network flows, this work depicted the multiple types of flow creation and resilience mechanisms for mitigating unavailability of the SDN networks. Besides, a hybrid resilience mechanism for OpenFlow networks and a scheme to mitigate *Packet-In* events generated from the controller to the OpenFlow stateful applications caused by multiple *Packet-In* packets sent from OpenFlow switches. Here proposed without requiring changes in the specification or the OpenFlow switches, the results of adopting the proposed scheme indicate that a stateful application can keep the internal state consistency concerning *Packet-In* events.

Furthermore, this work also extensively discussed the limitation of the OpenFlow technology, classified previous works in the management of flows and the inter-domain routing environment, the mechanisms to exchange control information among SDN domains, the attractiveness of deploying the architecture in the current Internet, and future works. In conclusion, the results and the extensive discussion of the proposed architecture makes this work a step further in the literature of the mechanisms for managing Internet traffic.

REFERENCES

ADRICHEM, N. L. V.; ASTEN, B. J. V.; KUIPERS, F. a. Fast Recovery in Software-Defined Networks. *2014 Third European Workshop on Software Defined Networks*, p. 61–66, 2014. ISSN 2379-0350.

AGER, B.; CHATZIS, N.; FELDMANN, A.; SARRAR, N.; UHLIG, S.; WILLINGER, W. Anatomy of a large european IXP. *ACM SIGCOMM Computer Communication Review*, v. 42, n. 4, p. 163, 2012. ISSN 01464833.

AKYILDIZ, I. F.; LEE, A.; WANG, P.; LUO, M.; CHOU, W. A roadmap for traffic engineering in software defined networks. *Computer Networks*, Elsevier B.V., v. 71, p. 1–30, 2014. ISSN 13891286.

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys & Tutorials*, PP, n. 99, p. 1–1, 2015. ISSN 1553-877X.

AL-MUSAWI, B.; BRANCH, P.; ARMITAGE, G. BGP Anomaly Detection Techniques: A Survey. *IEEE Communications Surveys & Tutorials*, v. 19, n. 1, p. 1–20, 2016.

ALIMI, R.; PENNO, R.; YANG, Y.; KIESEL, S.; PREVIDI, S.; ROOME, W.; SHALUNOV, S.; WOUNDY, R. Application-Layer Traffic Optimization (ALTO) Protocol Applications. 2014.

ALSHAMRANI, H.; GHITA, B. IP prefix hijack detection using BGP connectivity monitoring. *IEEE International Conference on High Performance Switching and Routing, HPSR*, v. 2016-July, p. 35–41, 2016. ISSN 23255609.

ANDO, S.; NAKAO, A. OpenFlow transparent custom action extension by using Packet-In and click packet processing. *Proceedings - Asia-Pacific Conference on Communications, APCC 2016*, p. 274–280, 2016. ISSN 19440375.

BANNOUR, F.; SOUIHI, S.; MELLOUK, A. Distributed SDN Control: Survey, Taxonomy and Challenges. *IEEE Communications Surveys and Tutorials*, n. c, p. 1–25, 2017. ISSN 1553877X.

BENNESBY, R.; MOTA, E. A survey on approaches to reduce BGP interdomain routing convergence delay on the Internet. *IEEE Communications Surveys and Tutorials*, n. c, 2017. ISSN 1553877X.

BENNESBY, R.; MOTA, E.; FONSECA, P.; PASSITO, A. Innovating on interdomain routing with an inter-SDN component. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, n. Ic, p. 131–138, 2014. ISSN 1550445X.

BIANCHI, G.; BONOLA, M.; CAPONE, A.; CASCONE, C. OpenState: Programming Platform-independent Stateful OpenFlow Applications Inside the Switch. *ACM SIGCOMM Computer Communication Review*, v. 44, n. 2, p. 44–51, 2014. ISSN 01464833.

BOSSHART, P.; VARGHESE, G.; WALKER, D.; DALY, D.; GIBB, G.; IZZARD, M.; MCKEOWN, N.; REXFORD, J.; SCHLESINGER, C.; TALAYCO, D.; VAHDAT, A. P4: Programming Protocol-Independent Packet Processors. *ACM SIGCOMM Computer Communication Review*, v. 44, n. 3, p. 87–95, 2014. ISSN 01464833.

CAIDA. Center for Applied Internet Data Analysis - CAIDA. *http://data.caida.org/datasets/as-relationships/serial-2/*, 2016. Available from Internet: <www.caida.org>.

CAIDA. AS Rank: AS Ranking - CAIDA. *http://as-rank.caida.org/*, 2017. Available from Internet: <http://as-rank.caida.org/>.

CARDONA, J. C.; VISSICCHIO, S.; LUCENTE, P.; FRANCOIS, P. "I Can't Get No Satisfaction": Helping Autonomous Systems Identify Their Unsatisfied Inter-domain Interests. *IEEE Transactions on Network and Service Management*, v. 13, n. 1, p. 43 – 57, 2016. ISSN 19324537.

CASCONE, C.; POLLINI, L.; SANVITO, D.; CAPONE, A.; SANSO, B. SPIDER: Fault resilient SDN pipeline with recovery delay guarantees. *IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conference and Workshops: Software-Defined Infrastructure for Networks, Clouds, IoT and Services*, p. 295–302, 2016.

CHANDA, A.; WESTPHAL, C. Content Based Traffic Engineering in Software Defined Information Centric Networks. *INFOCOM, 2013. 32nd IEEE International Conference on Computer Communications*, p. 3397–3402, 2013.

CHANDRA, R.; TRAINA, P. BGP Communities Attribute - RFC 1997. *Network Working Group*, p. 1–5, 1996. Available from Internet: <https://tools.ietf.org/pdf/rfc1997.pdf>.

CHEN, C.; LI, B.; LIN, D.; LI, B. Software-Defined Inter-Domain Routing Revisited. *IEEE ICC 2016 - Next-Generation Networking and Internet Symposium Software-Defined*, 2016.

CHEN, Z.; BI, J.; FU, Y.; WANG, Y.; XU, A. MLV: A multi-dimension routing information exchange mechanism for inter-domain SDN. *Proceedings - International Conference on Network Protocols, ICNP*, v. 2016-March, p. 438–445, 2016. ISSN 10921648.

CHIANG, M.; LOW, S. H.; CALDERBANK, A. R.; DOYLE, J. C. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, v. 95, n. 1, p. 255–312, 2007. ISSN 00189219.

CHOWDHURY, M. K. N.; BOUTABA, R. Network virtualization: state of the art and research challenges. *IEEE Communications Magazine*, v. 47, n. 7, p. 20–26, 2009. ISSN 0163-6804.

CHOWDHURY, N. M. M. K.; BOUTABA, R. A survey of network virtualization. *Computer Networks*, Elsevier B.V., v. 54, n. 5, p. 862–876, 2010. ISSN 13891286.

CVJETIC, A.; SMILJANIC, A. Improving BGP protocol to advertise multiple routes for the same destination prefix. *IEEE Communications Letters*, v. 18, n. 1, p. 106–109, 2014. ISSN 10897798.

D. Katz and D. Ward. BFD for IPv4 and IPv6 Single Hop. *RFC 5881*, n. 1, p. 1–5, 2010. ISSN 0717-6163.

D. Katz and D. Ward. Bidirectional Forwarding Detection. *RFC 5880*, v. 53, p. 160, 2010. ISSN 1098-6596.

DATTA, A.; RASTOGI, A.; BARMAN, O. R.; D'MELLO, R.; ABUZAGHLEH, O. An Approach for Implementation of Artificial Intelligence in Automatic Network Management and Analysis. *Online Engineering & Internet of Things*, v. 22, p. 901–909, 2018.

DEUS, M. A. de; CARVALHO, P. H.; LEITE, J. P. Internet Traffic Engineering: Understanding the Specialist Decision-Making Process for Traffic Management in South America. *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, p. 37–41, 2014.

DEUS, M. A. de; CARVALHO, P. H.; LEITE, J. P. Internet capacity: optimizing autonomous system inbound traffic using specialist knowledge as support for decision-making. *Annales des Telecommunications/Annals of Telecommunications*, v. 70, n. 7-8, p. 331–343, 2015. ISSN 19589395.

ESTEVES, R. P.; GRANVILLE, L. Z.; BOUTABA, R. On the management of virtual networks. *IEEE Communications Magazine*, v. 51, n. 7, p. 80–88, 2013. ISSN 01636804.

FADLULLAH, Z.; TANG, F.; MAO, B.; KATO, N.; AKASHI, O.; INOUE, T.; MIZUTANI, K. State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. *IEEE Communications Surveys & Tutorials*, n. c, p. 1–1, 2017. ISSN 1553-877X.

FEAMSTER, N.; BALAKRISHNAN, H.; REXFORD, J.; SHAIKH, A.; MERWE, J. van der. The case for separating routing from routers. *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture - FDNA '04*, p. 5, 2004. ISSN 158113942X.

FEAMSTER, N.; FEAMSTER, N.; BALAKRISHNAN, H.; BALAKRISHNAN, H. Detecting BGP configuration faults with static analysis. *Proc. Networked Systems Design and Implementation*, p. 49–56, 2005.

FEAMSTER, N.; REXFORD, J.; SHENKER, S.; CLARK, R.; HUTCHINS, R.; LEVIN, D.; BAILEY, J. SDX: A software-defined Internet exchange. *Open Networking Summit*, p. 2–3, 2013.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to SDN: An Intellectual History of Programmable Networks. *ACM SIGCOMM Computer Communication Review*, v. 44, n. 2, p. 87–98, 2014. ISSN 01464833.

FERNANDEZ, M. P. Comparing OpenFlow controller paradigms scalability: Reactive and proactive. In: *International Conference on Advanced Information Networking and Applications, AINA*. [S.l.: s.n.], 2013. p. 1009–1016. ISBN 9780769549538. ISSN 1550445X.

FONSECA, P.; MOTA, E. A Survey on Fault Management in Software-Defined Networks. *IEEE Communications Surveys and Tutorials*, n. c, p. 1–39, 2017. ISSN 1553877X.

FUJINOKI, H. Multi-Path BGP (MBGP): A Solution for Improving Network Bandwidth Utilization and Defense against Link Failures in Inter-Domain Routing. In: *16th IEEE International Conference on Networks*. [S.l.: s.n.], 2008. p. 6. ISBN 9781424438051.

GÄMPERLI, A.; KOTRONIS, V.; DIMITROPOULOS, X. Evaluating the effect of centralization on routing convergence on a hybrid BGP-SDN emulation framework. *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*, p. 369–370, 2014.

GANICHEV, I.; DAI, B.; GODFREY, P. B.; SHENKER, S. YAMR: Yet Another Multipath Routing Protocol. *ACM SIGCOMM Computer Communication Review*, v. 40, n. 5, p. 13, 2010. ISSN 01464833.

GAO, L. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, v. 9, n. 6, p. 733–745, 2001. ISSN 10636692.

GIOTSAS, V.; LUCKIE, M.; HUFFAKER, B. Inferring Complex AS Relationships. *Proceedings of the 2014 Conference on Internet Measurement Conference*, p. 23–29, 2014.

GODFREY, P. B.; CAESAR, M.; HAKEN, I.; SINGER, Y.; SHENKER, S.; STOICA, I. Stabilizing route selection in BGP. *IEEE/ACM Transactions on Networking*, v. 23, n. 1, p. 282–299, 2015. ISSN 10636692.

GUPTA, A.; JHA, R. K. A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access*, v. 3, n. c, p. 1206–1232, 2015. ISSN 21693536.

GUPTA, A.; MACDAVID, R.; BIRKNER, R.; CANINI, M.; FEAMSTER, N.; REXFORD, J.; VANBEVER, L. An Industrial-Scale Software Defined Internet Exchange Point. *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, p. 1–14, 2016.

H. Gredler, E.; MEDVED, J.; PREVIDI, S.; FARREL, A.; RAY, S. North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP. *Internet Engineering Task Force (IETF)*, p. 1–48, 2016. Available from Internet: <https://tools.ietf.org/pdf/rfc7752.pdf>.

HAKIRI, A.; GOKHALE, A.; BERTHOU, P.; SCHMIDT, D. C.; GAYRAUD, T. Software-defined networking: Challenges and research opportunities for future internet. *Computer Networks*, Elsevier B.V., v. 75, n. PartA, p. 453–471, 2014. ISSN 13891286.

HE, J.; REXFORD, J. Toward internet-wide multipath routing. *IEEE Network*, v. 22, n. 2, p. 16–21, 2008. ISSN 08908044.

HELLER, B.; SHERWOOD, R.; MCKEOWN, N. The controller placement problem. *ACM SIGCOMM Computer Communication Review*, v. 42, n. 4, p. 473, 2012. ISSN 01464833.

HONG, C.-Y.; KANDULA, S.; MAHAJAN, R.; ZHANG, M.; GILL, V.; NANDURI, M.; WATTENHOFER, R. Achieving high utilization with software-driven WAN. *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13*, p. 15, 2013. ISSN 0146-4833.

HUANG, T.; YU, F. R.; ZHANG, C.; LIU, J.; ZHANG, J.; LIU, Y. A Survey on Large-Scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges. *IEEE Communications Surveys and Tutorials*, v. 19, n. 2, p. 891–917, 2017. ISSN 1553877X.

IPERF. iperf - perform network throughput tests. *http://iperf.sourceforge.net/*, 2016. Available from Internet: <http://iperf.sourceforge.net/>.

IRR. *Internet Routing Registry*. 2017. Available from Internet: <http://www.irr.net/>.

J. Heitz, E.; J. Snijders, E.; PATEL, K.; BAGDONAS, I.; HILLIARD, N. BGP Large Communities Attribute Abstract. 2017.

JACQUEMART, Q.; URVOY-KELLER, G.; BIERSACK, E. A longitudinal study of BGP MOAS prefixes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 8406 LNCS, p. 127–138, 2014. ISSN 16113349.

JAIN, S.; ZHU, M.; ZOLLA, J.; HÖLZLE, U.; STUART, S.; VAHDAT, A.; KUMAR, A.; MANDAL, S.; ONG, J.; POUTIEVSKI, L.; SINGH, A.; VENKATA, S.; WANDERER, J.; ZHOU, J. B4. *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13*, p. 3, 2013. ISSN 0146-4833.

KANG, N.; LIU, Z.; REXFORD, J.; WALKER, D. Optimizing the "one big switch" abstraction in software-defined networks. *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies - CoNEXT '13*, p. 13–24, 2013.

KANIZO, Y.; HAY, D.; KESLASSY, I. Palette: Distributing tables in software-defined networks. *Proceedings - IEEE INFOCOM*, p. 545–549, 2013. ISSN 0743166X.

KEVIN, B.; TONI, F.; PATRICK, M.; JENNNIFER, P. A survey of BGP security - Issues and solutions. *Vol. V, No. N*, V, n. April, p. 1–35, 2010. ISSN 0018-9219.

KHIYAITA, A.; BAKKALI, H. E. L.; ZBAKH, M.; KETTANI, D. E. Load balancing cloud computing: State of art. *Network Security and Systems (JNS2)*, v. 1, n. 1, p. 106 – 109, 2012.

KING, T.; DIETZEL, C.; SNIJDERS, J.; DOERING, G.; HANKINS, G. BLACKHOLE Community This. 2016.

KOERNER, M. Multiple Service Load-Balancing with OpenFlow. *IEEE 13th International Conference on High Performance Switching and Routing Multiple*, p. 210–214, 2012.

KOTRONIS, V. *Centralizing Routing Control Across Domains: Architectural Approach and Prominent Use Cases*. 1–266 p. Tese (Doutorado) — University of Athens, 2015.

KOTRONIS, V.; DIMITROPOULOS, X.; AGER, B. Outsourcing the routing control logic: better internet routing based on SDN principles. *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, p. 55–60, 2012.

KOTRONIS, V.; DIMITROPOULOS, X.; KLÖTI, R.; AGER, B.; GEORGOPOULOS, P.; SCHMID, S. Control Exchange Points: Providing QoS-enabled End-to-End Services via SDN-based Inter-domain Routing Orchestration. *Linx*, v. 2429, n. 1093, p. 2443, 2014.

KOTRONIS, V.; GAMPERLI, A.; DIMITROPOULOS, X. Routing centralization across domains via SDN: A model and emulation framework for BGP evolution. *Computer Networks*, v. 92, p. 227–239, 2015. ISSN 13891286.

KREUTZ, D.; RAMOS, F. M.; VERISSIMO, P. Towards secure and dependable software-defined networks. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, p. 55, 2013.

KREUTZ, D.; RAMOS, F. M. V.; VERÍSSIMO, P. E.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14 – 76, 2015. ISSN 0018-9219.

KUSHMAN, N.; KANDULA, S.; KATABI, D.; MAGGS, B. M. R-BGP: Staying Connected In a Connected World. *NSDI'07 Proceedings of the 4th USENIX conference on Networked systems design & implementation*, p. 14, 2007.

LABOVITZ, C.; IEKEL-JOHNSON, S.; MCPHERSON, D.; OBERHEIDE, J.; JAHANIAN, F. Internet inter-domain traffic. *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM '10*, p. 75, 2010. ISSN 1450302017.

LEE, S. J.; BANERJEE, S.; SHARMA, P.; YALAGANDULA, P.; BASU, S. Bandwidth-aware routing in overlay networks. *Proceedings - IEEE INFOCOM*, p. 2405–2413, 2008. ISSN 0743166X.

LI, M.; LUKYANENKO, A.; OU, Z.; YLA-JAASKI, A.; TARKOMA, S.; COUDRON, M.; SECCI, S. Multipath Transmission for the Internet: A Survey. *IEEE Communications Surveys and Tutorials*, v. 18, n. 4, p. 2887–2925, 2016. ISSN 1553877X.

LIAO, Y.; GAO, L.; GUERIN, R.; ZHANG, Z.-L. Reliable interdomain routing through multiple complementary routing processes. *Proceedings of the 2008 ACM CoNEXT Conference on - CONEXT '08*, p. 1–6, 2008.

LIN, P.; BI, J.; CHEN, Z.; WANG, Y.; HU, H.; XU, A. WE-bridge: West-east bridge for SDN inter-domain network peering. *Proceedings - IEEE INFOCOM*, p. 111–112, 2014. ISSN 0743166X.

LIN, P.; HART, J.; KRISHNASWAMY, U. Seamless interworking of SDN and IP. *SIGCOMM '13 Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, v. 43, n. 4, p. 475–476, 2013. ISSN 0146-4833.

LIN, Y. D.; TENG, H. Y.; HSU, C. R.; LIAO, C. C.; LAI, Y. C. Fast failover and switchover for link failures and congestion in software defined networks. *2016 IEEE International Conference on Communications, ICC 2016*, 2016. ISSN 1098-6596.

LIU, H. H.; KANDULA, S.; MAHAJAN, R.; ZHANG, M.; GELERNTER, D. Traffic Engineering with Forward Fault Correction. *Proceedings of the 2014 ACM Conference on SIGCOMM*, p. 527–538, 2014. ISSN 19435819.

LIU, X.; MOHANRAJ, S.; PIORO, M.; MEDHI, D. Multipath Routing From a Traffic Engineering Perspective: How Beneficial is It? *22nd IEEE ICNP, Raleigh, North Carolina, USA*, n. 2011, p. 21–23, 2014. ISSN 10921648.

LUCKIE, M.; HUFFAKER, B.; DHAMDHERE, A.; GIOTSAS, V.; CLAFFY, K. AS relationships, customer cones, and validation. *Proceedings of the 2013 conference on Internet measurement conference - IMC '13*, p. 243–256, 2013.

LUDWIG, C. Traffic engineering with BGP. *Seminar "Internet Routing" , Technical University Berlin*, p. 1–10, 2009.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, v. 38, n. 2, p. 69, 2008. ISSN 01464833.

MENDIOLA, A.; ASTORGA, J.; JACOB, E.; HIGUERO, M. A Survey on the Contributions of Software-Defined Networking to Traffic Engineering. *IEEE Communications Surveys and Tutorials*, v. 19, n. 2, p. 918–953, 2017. ISSN 1553877X.

MININET. *Mininet - An Instant Virtual Network on your Laptop (or other PC)*. 2016. Available from Internet: <http://mininet.org/>.

MIZUYAMA, K.; TAENAKA, Y.; TSUKAMOTO, K. Estimation Based Adaptable Flow Aggregation Method for Reducing Control Traffic on Software Defined Wireless Networks. *The 8th International Workshop on Information Quality and Quality of Service for Pervasive Computing*, 2017.

MUQADDAS, A. S.; BIANCO, A.; GIACCONE, P.; MAIER, G. Inter-controller Traffic in ONOS Clusters for SDN Networks. *IEEE ICC 2016 - Next-Generation Networking and Internet Symposium*, 2016.

MUQADDAS, A. S.; GIACCONE, P.; BIANCO, A.; MAIER, G. Inter-controller Traffic to Support Consistency in ONOS Clusters. *IEEE Transactions on Network and Service Management*, v. 4537, n. c, p. 1–14, 2017. ISSN 19324537.

NARAYANAN, A. A Survey on BGP Issues and Solutions. 2009. Available from Internet: <http://arxiv.org/abs/0907.4815>.

NASCIMENTO, M. R.; ROTHENBERG, C. E.; SALVADOR, M. R.; CORRÊA, C. N. A.; LUCENA, S. C. de; MAGALHÃES, M. F. Virtual routers as a service: The RouteFlow Approach Leveraging Software-Defined Networks. *Proceedings of the 6th International Conference on Future Internet Technologies - CFI '11*, n. June 2016, p. 34, 2011.

NENCIONI, G.; HELVIK, B. E.; GONZALEZ, A. J.; HEEGAARD, P. E.; KAMISINSKI, A. Availability Modelling of Software-Defined Backbone Networks. *Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN-W 2016*, p. 105–112, 2016.

NGUYEN, X.-n.; SAUCEZ, D.; BARAKAT, C.; TURLETTI, T.; SOPHIA, I.; MÉDITERRANÉE, A. Optimizing Rules Placement in OpenFlow Networks : Trading Routing for Better Efficiency. *HotSDN 2014*, n. HotSDN, p. 127–132, 2014.

NGUYEN, X.-n.; SAUCEZ, D.; BARAKAT, C.; TURLETTI, T. Rules Placement Problem in OpenFlow Networks: A Survey. *IEEE Communications Surveys & Tutorials*, v. 18, n. 2, p. 1273–1286, 2016.

Ning Wang, Kin Hon Ho, George Pavlou, M. H. An Overview of Routing Optimization for Internet Traffic Engineering. *IEEE Communications Surveys*, v. 10, n. 1, p. 36–56, 2008.

Niven-Jenkins, B and Brungard, D and Betts, M and Sprecher, N and Ueno, S. Requirements of an MPLS transport profile. *https://tools.ietf.org/html/draft-ietf-mpls-tp-requirements-10*, 2009. Available from Internet: <https://tools.ietf.org/html/draft-ietf-mpls-tp-requirements-10>.

NUNES, B. A. A.; MENDONCA, M.; NGUYEN, X. N.; OBRACZKA, K.; TURLETTI, T. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, v. 16, n. 3, p. 1617–1634, 2014. ISSN 1553877X.

ONF. *ONF Overview - The Open Networking Foundation.* 2017. Available from Internet: <https://www.opennetworking.org/about/onf-overview>.

ONOS. A new carrier-grade SDN network operation system designed for high availability,performance, scale-out. *http://onosproject.org/*, 2017. Available from Internet: <http://onosproject.org/>.

OPENNETWORKING. *OpenFlow Switch Specification, version 1.5.0.* 2014. 1–205 p. Available from Internet: <https://www.opennetworking.org>.

OPENVSWITCH. Open vSwitch. *http://openvswitch.org/*, 2016. Available from Internet: <http://openvswitch.org/>.

PFAFF, B.; LANTZ, B.; HELLER, B.; BARKER, C.; COHN, D.; TALAYCO, D.; ERICKSON, D.; CRABBE, E.; GIBB, G.; APPENZELLER, G.; TOURRILHES, J.; PETTIT, J.; YAP, K.; POUTIEVSKI, L.; CASADO, M.; TAKAHASHI, M.; KOBAYASHI, M.; M, N.; KIS, Z. L. OpenFlow 1.1 Specification. *Open Networking Foundation*, p. 1–56, 2011. Available from Internet: <http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf>.

PFAFF, B.; LANTZ, B.; HELLER, B.; BARKER, C.; COHN, D.; CASADO, M. OpenFlow Switch Specification - 1.3 version. *Open Networking Foundation*, v. 0, p. 0–105, 2012.

POESE, I.; FRANK, B.; AGER, B.; SMARAGDAKIS, G.; FELDMANN, A. Improving content delivery using provider-aided distance information. *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, p. 22, 2010.

POESE, I.; FRANK, B.; SMARAGDAKIS, G.; UHLIG, S.; FELDMANN, A.; MAGGS, B. Enabling content-aware traffic engineering. *Computer Communication Review*, v. 42, n. 5, p. 22–28, 2012. ISSN 01464833 (ISSN).

POTAROO.NET. *Advertised AS Count.* 2016. Available from Internet: <http://bgp.potaroo.net/bgprpts/rva-index.html>.

QIN, D.; YANG, J.; LIU, Z.; WANG, H.; ZHANG, B.; ZHANG, W. AMIR: Another multipath interdomain routing. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, p. 581–588, 2012. ISSN 1550445X.

QIU, J.; WANG, F.; GAO, L. BGP rerouting solutions for transient routing failures and loops. *Proceedings - IEEE Military Communications Conference MILCOM*, 2007.

QUAGGA. Quagga Routing Suite. *http://www.nongnu.org/quagga/*, 2017. Available from Internet: <http://www.nongnu.org/quagga/>.

RAGHAVAN, B.; CASADO, M.; KOPONEN, T.; RATNASAMY, S.; GHODSI, A.; SHENKER, S. Software-defined internet architecture. *Proceedings of the 11th ACM Workshop on Hot Topics in Networks - HotNets-XI*, p. 43–48, 2012.

RAO, A.; LEGOUT, A.; LIM, Y.-s.; TOWSLEY, D.; BARAKAT, C.; DABBOUS, W. Network characteristics of video streaming traffic. *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies (CoNEXT)*, p. 1–12, 2011.

REKHTER, Y.; LI, T.; HARES, S. A Border Gateway Protocol 4 (BGP-4). *Network Working Group - Request for Comments: 4271*, 2006. ISSN 1098-6596.

ROTHENBERG, C. E.; NASCIMENTO, M. R.; SALVADOR, M. R.; CORRÊA, C. N. A.; Cunha de Lucena, S.; RASZUK, R. Revisiting routing control platforms with the eyes and muscles of software-defined networking. *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, p. 13, 2012.

RYU. A component-based software defined networking framework - Ryu. *https://osrg.github.io/ryu/*, 2016. Available from Internet: <https://osrg.github.io/ryu/>.

SALTZER, J. H.; REED, D. P.; CLARK, D. D. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, v. 2, n. 4, p. 277–288, 1984. ISSN 07342071.

SCOTT-HAYWARD, S.; NATARAJAN, S.; SEZER, S. A Survey of Security in Software Defined Networks. *IEEE Communications Surveys & Tutorials*, PP, n. 99, p. 1–33, 2015. ISSN 1553-877X.

SCUDDER, J.; FERNANDO, R.; STUART, S. BGP Monitoring Protocol (BMP). *Internet Engineering Task Force*, p. 1–27, 2016. Available from Internet: <https://tools.ietf.org/html/rfc7854>.

SHAO, W.; DEVIENNE, F.; IANNONE, L.; ROUGIER, J.-L. On the use of BGP communities for fine-grained inbound traffic engineering. 2015. Available from Internet: <http://arxiv.org/abs/1511.08336>.

SHAO, W.; DEVIENNE, F.; IANNONE, L.; ROUGIER, J.-L. On the Use of BGP Communities for Fine-grained Inbound Traffic Engineering. *arXiv preprint arXiv:1511.08336*, 2015. Available from Internet: <http://arxiv.org/abs/1511.08336>.

SHARMA, S.; STAESSENS, D.; COLLE, D.; PICKAVET, M.; DEMEESTER, P. Enabling Fast Failure Recovery in OpenFlow Networks. *IEEE Design of Reliable Communication Networks*, n. October, p. 164–171, 2011.

SHARMA, S.; STAESSENS, D.; COLLE, D.; PALMA, D.; GONCALVES, J.; FIGUEIREDO, R.; MORRIS, D.; PICKAVET, M.; DEMEESTER, P. Implementing Quality of Service for the Software Defined Networking Enabled Future Internet. *2014 Third European Workshop on Software Defined Networks*, p. 49–54, 2014.

SILVA, W. J. A. Performance Evaluation of Flow Creation Inside an OpenFlow Network. In: *XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT2017*. São Pedro, SP: SBrT, 2017. p. 102–106.

SILVA, W. J. A. An Architecture to Manage Incoming Traffic of Inter-Domain Routing Using OpenFlow Networks. *Information*, v. 9, n. 4, p. 1–23, 2018.

SILVA, W. J. A. Avoiding Inconsistency in OpenFlow Stateful Applications Caused by Multiple Flow Requests. *International Conference on Computing, Networking and Communications (ICNC)*, p. 543–548, 2018.

SILVA, W. J. A. Make Flows Great Again : A Hybrid Resilience Mechanism for OpenFlow Networks. *Information*, v. 9, n. 6, p. 1–19, 2018.

SILVA, W. J. A.; DIAS, K. L.; SADOK, D. F. H. A Performance Evaluation of Software Defined Networking Load Balancers Implementations. In: *International Conference on Information Networking (ICOIN)*. [S.l.: s.n.], 2017. ISBN 9781509051243.

SILVA, W. J. A.; Djamel Fawzi Hadj Sadok. A Survey on Efforts to Evolve the Control Plane of Inter-Domain Routing. *Information*, v. 9, n. 5, p. 1–26, 2018.

SILVA, W. J. A.; SADOK, D. F. H. Control Inbound Traffic: Evolving the Control Plane Routing System with Software Defined Networking. In: *18th International Conference on High Performance Switching and Routing (HPSR)*. [S.l.: s.n.], 2017. ISBN 9781509028399.

SINGH, S.; DAS, T.; JUKAN, A. A Survey on Internet Multipath Routing and Provisioning. *IEEE Communications Surveys & Tutorials*, v. 17, n. July, p. 1–1, 2015. ISSN 1553-877X.

SMITH, P. *BGP Best Current Practices*. 2017. Available from Internet: <https://nsrc.org/workshops/2017/apricot2017/bgp/bgp/preso/05-BGP-BCP.pdf>.

SNIJDERS, J.; HEITZ, J.; SCUDDER, J. BGP Administrative Shutdown Communication. 2017.

SOMANI, G.; GAUR, M. S.; SANGHI, D.; CONTI, M.; BUYYA, R. DDoS Attacks in Cloud Computing: Issues, Taxonomy, and Future Directions. v. 1, n. 1, 2015. Available from Internet: <http://arxiv.org/abs/1512.08187>.

THAI, P.; De Oliveira, J. C. Decoupling policy from routing with software defined interdomain management: Interdomain routing for SDN-based networks. *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, 2013. ISSN 10952055.

TOPOLOGY-ZOO. *The internet topology zoo*. 2017. Available from Internet: <http://www.topology-zoo.org/dataset.html>.

Van Beijnum, I.; CROWCROFT, J.; VALERA, F.; BAGNULO, M. Loop-freeness in multipath BGP through propagating the longest path. *Proceedings - 2009 IEEE International Conference on Communications Workshops, ICC 2009*, 2009. ISSN 2164-7038.

VIRTUALBOX. Virtualization – Oracle VM VirtualBox. *https://www.virtualbox.org/wiki/Virtualization*, 2016. Available from Internet: <https://www.virtualbox.org/wiki/Virtualization>.

VISHNOI, A.; PODDAR, R.; MANN, V.; BHATTACHARYA, S. Effective Switch Memory Management in OpenFlow Networks. *DEBS '14- Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, 2014.

VISHNOI, A.; PODDAR, R.; MANN, V.; BHATTACHARYA, S. Effective Switch Memory Management in OpenFlow Networks. *8th ACM International Conference on Distributed Event-Based Systems*, p. 177–188, 2014.

WALTON, D.; RETANA, A.; CHEN, E.; SCUDDER, J. Advertisement of Multiple Paths in BGP - RFC 7911. *Internet Engineering Task Force (IETF)*, p. 1–8, 2016.

WANG, T.; SHI, X.; YIN, X.; WANG, Z. Quantifying the Propagation Behavior of BGP Routing Update. *Bulletin of Networking, Computing, Systems, and Software*, v. 4, n. 1, p. 18–20, 2015.

WANG, Y.; BI, J.; LIN, P.; LIN, Y.; ZHANG, K. SDI: a multi-domain SDN mechanism for fine-grained inter-domain routing. *Annales des Telecommunications/Annals of Telecommunications*, Annals of Telecommunications, v. 71, n. 11-12, p. 625–637, 2016. ISSN 19589395.

WANG, Y.; BI, J.; ZHANG, K.; WU, Y. A Framework for Fine-Grained Inter-Domain Routing Diversity Via SDN. *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, p. 751–756, 2016.

WICHTLHUBER, M.; REINECKE, R.; HAUSHEER, D. An SDN-based CDN/ISP collaboration architecture for managing high-volume flows. *IEEE Transactions on Network and Service Management*, v. 12, n. 1, p. 48–60, 2015. ISSN 19324537.

WIRESHARK. *Wireshark - Network Protocol Analyzer*. 2017. Available from Internet: <https://www.wireshark.org/>.

WOLF, T.; GRIFFIOEN, J.; CALVERT, K.; DUTTA, R.; ROUSKAS, G.; NAGURNEY, A. ChoiceNet : Toward an Economy Plane for the Internet. *ACM SIGCOMM Computer Communication Review*, v. 44, n. 3, p. 58–65, 2014. ISSN 19435819.

XU, W.; REXFORD, J. MIRO : Multi-path Interdomain ROuting. *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, p. 171–182, 2006. ISSN 01464833.

Y. Rekhter, T.J. Watson, T. L. A Border Gateway Protocol 4 (BGP-4). *Request for Comments: 1771*, 1995.

YANG, X.; CLARK, D.; BERGER, A. W. NIRA: A new inter-domain routing architecture. *IEEE/ACM Transactions on Networking*, v. 15, n. 4, p. 775–788, 2007. ISSN 10636692.

YANNUZZI, M.; MASIP-BRUIN, X.; BONAVENTURE, O. Open issues in interdomain routing: A survey. *IEEE Network*, v. 19, n. 6, p. 49–56, 2005. ISSN 08908044.

ZARGAR, S. T.; JOSHI, J.; TIPPER, D. A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks. *IEEE Communications Surveys and Tutorials*, v. 15, n. 4, p. 2046–2069, 2013. ISSN 1553877X.

ZHANG, X.; CHENG, Z.; LIN, R.; HE, L.; YU, S.; LUO, H. Local Fast Reroute with Flow Aggregation in Software Defined Networks. *IEEE Communications Letters*, v. 7798, n. c, p. 1–4, 2016.

ZHU, D.; GRITTER, M.; CHERITON, D. R. Feedback based routing. *ACM SIGCOMM Computer Communication Review*, v. 33, n. 1, p. 71–76, 2003. ISSN 01464833.